Lorenzo Abruzzi Dias

Polarização de Canal e Códigos Polares

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica Orientador: Prof. Dr. Bartolomeu Ferreira Uchôa-Filho

Florianópolis, 2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Dias, Lorenzo Abruzzi
Polarização de canal e códigos polares / Lorenzo Abruzzi Dias ; orientador, Bartolomeu Ferreira Uchôa-Filho, 2018.
59 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2018.

Inclui referências.

1. Engenharia Eletrônica. 2. Polarização de canal. 3. Códigos polares. 4. Cancelamento sucessivo. I. Uchôa-Filho, Bartolomeu Ferreira. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Lorenzo Abruzzi Dias

Polarização de canal e códigos polares

Este trabalho foi julgado adequado para a obtenção do Título de Bacharel em Engenharia Eletrônica e aprovado em sua forma final pela Banca Examinadora.

Florianópolis, 29 de Janeiro de 2018

Prof. Jefferson Luiz Brum Marques, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Bartolomeu Ferreira Uchôa-Filho, Dr.
Orientador
Universidade Federal de Santa Catarina

Prof. Danilo Silva, Dr.
Universidade Federal de Santa Catarina

Eng. Gustavo Kasper Facenda

Este trabalho é dedicado aos meus queridos pais, ao meu querido irmão e aos meus queridos colegas de classe.

AGRADECIMENTOS

A Universidade Federal de Santa Catarina, pela oportunidade de fazer o curso e pelo ambiente criativo e amigável que proporcionou.

Ao professor Bartolomeu Ferreira Uchôa-Filho, pela orientação, apoio e confiança.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

Aos meus pais e meu irmão, pelo amor, incentivo e apoio incondicional.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

A principal função de um sistema de comunicação é reproduzir, exatamente ou de forma aproximada, uma informação proveniente de outro ponto diferente.

(Claude Shannon, 1948)

RESUMO

Os códigos polares foram introduzidos por Arikan em 2009. Eles são os primeiros códigos de correção de erros que comprovadamente alcançam a capacidade de qualquer canal binário discreto sem memória(B-DMC) simétrico com algoritmos de codificação e de decodificação muito eficientes, usando a técnica de polarização de canal. A construção do código é baseada em uma concatenação recursiva múltipla de um código curto que transforma o canal físico em canais externos virtuais. Quando o número de recursões se torna grande, os canais virtuais tendem a ter alta confiabilidade ou baixa confiabilidade, criando assim uma polarização entre eles, e os bits de dados são alocados aos canais mais confiáveis. Para a decodificação é utilizada a técnica de cancelamento sucessivo, a qual gera uma estimativa do bit de entrada observando a saída do canal e utilizando as estimativas dos bits anteriores. Notavelmente, os códigos polares possuem baixa complexidade de codificação e descodificação $O(n\ log\ n)$, o que os torna atraentes para diversas aplicações.

Palavras-chave: Polarização de canal. Códigos polares. Codificação. Descodificação. Cancelamento sucessivo.

ABSTRACT

Polar codes were introduced by Arikan in 2009. They are the first error correction codes that have proven to be capable to achieve the channel capacity of any symmetric binary discrete memoryless channel with very efficient coding and decoding algorithms using the channel polarization technique. The code construction is based on a multiple recursive concatenation of a short code that transforms the physical channel into virtual external channels. When the number of recursions becomes large, the virtual channels tend to have high reliability or low reliability, thus creating a polarization between them, and the data bits are allocated to the most reliable channels. For decoding, the successive cancellation technique is used, which generates an estimate of the input bit by observing the output of the channel and using the estimates of the previous bits. Notably, the polar codes have low coding and decoding complexity O (n log n), which makes them attractive for many applications.

Keywords: Channel polarization. Polar codes. Coding. Decoding. Successive cancellation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

B-DMC - canal binário discreto sem memória (binary discrete memoryless channel)
BEC - canal binário com apagamento (binary erasure channel)
CN - nó de verificação (check node)
VN - nó variável (variable node)
LDPC - códigos de verificação de paridade de baixa densidade (low-density parity-check code)
LLR - razão logarítmica de verossimilhança (log-likelihood ratio)
LL - razão de verossimilhança (likelihood ratio)
LR - razão de verossimilhança (likelihood ratio)
AWGN - ruído aditivo Gaussiano branco (additive white gaussian noise)
BPSK - modulação por deslocamento de fase binário (binary phase shift keying)
SC - cancelamento sucessivo (successive cancellation)
SNR - razão sinal-ruído (signal-to-noise ratio)
BP - propagação de “crenças” (belief propagation)

LISTA DE SÍMBOLOS

$\otimes$ Produto de Kronecker

$\oplus$ Soma em módulo 2

∞ Infinito

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de comunicação digital são onipresentes em nossas vidas diariamente. Existem muitos exemplos, como telefones celulares, televisão digital, conexões de internet sem fio, etc. Esses sistemas são compostospelos blocos apresentados na Figura 1.1.

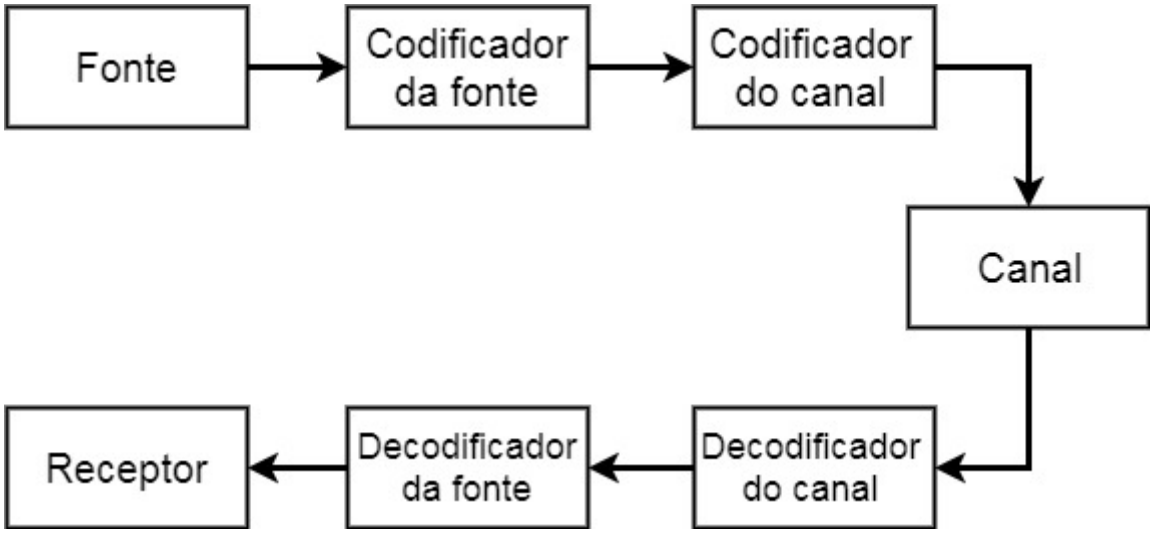


Figura 1.1: Diagrama de bloco geral de um sistema de comunicação

Os principais componentes de um sistema de comunicação são um codificador de fonte, para compressão de dados, um codificador de canal, que adiciona redundância, um canal de comunicação, que é por onde a informação é transmitida e cujo modelo probabilístico depende da aplicação e do meio, um decodificador de canal, que recupera os erros introduzidos pelo canal ruidoso, e um decodificador de fonte, que descomprime dados. Esta estrutura foi introduzida pela primeira vez por C. Shannon em 1948 em [5]. Neste trabalho, serão discutidos apenas sobre os blocos centrais, ou seja, os blocos de codificação e de decodificação de canal, mostrados na Figura 2.

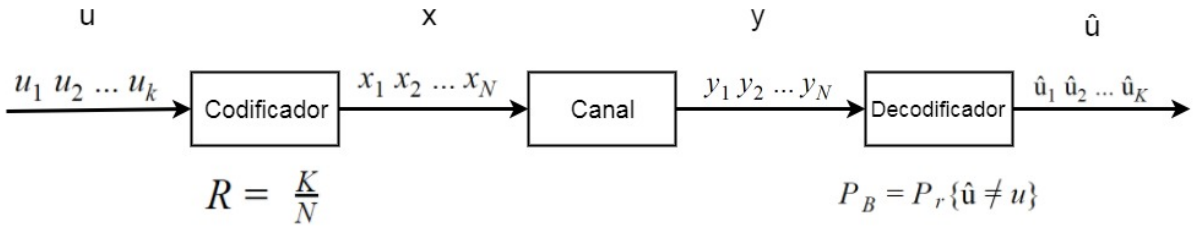


Figura 1.2: Diagrama de bloco de codificação de canal

O papel do codificador de canal é preservar a mensagem a ser transmitida através de um canal sujeito a ruído, distorção e interferência. Geralmente, deseja-se transmitir mensagens binárias, compostas por apenas dois símbolos, "0" e "1". Se

um 0 for enviado, é desejável ele seja recebido como um 0, mas pode acontecer de um 1 ser recebido, ou viceversa. Assim, para cada transmissão, o codificador de canal adiciona redundância e transforma o vetor inicial u = (u_1, u_2, ..., u_K) de comprimento K em um vetor c = (c_1, c_2, ..., c_N) de comprimento N, em que a fração R=K/N é chamada de taxa de código. Depois disso, c é mapeado em símbolos de modulação, resultando no vetor x. Existe uma probabilidade de o canal apresentar erros, de modo que ele mude um 0 para um 1 em uma mensagem transmitida. Dizemos então que ocorreu um erro. O decodificador de canal recebe um vetor y = (y_1, y_2, ..., y_N). O objetivo do decodificador é recuperar a entrada para o codificador de canal a partir da saída do canal, estimando u. A probabilidade de erro após o decodificador é chamada de probabilidade de erro de bloco P_B. Shannon mostrou que é possível transmitir dados digitais com alta confiabilidade, em canais ruidosos, codificando a mensagem com um código de correção de erros. O codificador de canal mapeia cada vetor de K bits em um vetor de N bits, chamado palavra-código. Assim, deve-se introduzir redundância para se conseguir uma comunicação segura e confiável sobre canais não confiáveis.

De acordo com Shannon, cada sistema de comunicação pode ser caracterizado por um parâmetro C, chamado de capacidade do canal, que é uma medida da quantidade de informação que se pode transmitir pelo canal de forma confiável [6]. Se R < C, existe um código com o comprimento N tal que a probabilidade de erro tende a zero se $N \rightarrow \infty$. Embora seu teorema mostre a existência de um bom código de canal, o principal objetivo depois de 1948 ainda é encontrar modos de codificação praticáveis, que possam alcançar a capacidade do canal em diferentes canais de comunicação.

Nas últimas décadas, com a descoberta de códigos turbo [7] e códigos de verificação de paridade de baixa densidade, LDPC [8], o limite de Shannon foi finalmente alcançado, mas apenas em casos particulares com códigos LDPC irregulares [9], e apenas para o canal binário com apagamento (BEC).

Os códigos polares foram introduzidos por Arikan em 2009 em [1]. Eles são a primeira classe de códigos de correção de erros que comprovadamente alcança a capacidade de qualquer canal binário discreto sem memória(B-DMC) simétrico com algoritmos de codificação e de decodificação muito eficientes, usando a técnica de polarização de canal. Este trabalho de conclusão de curso apresenta um estudo sobre polarização de canal e códigos polares.

2 INTRODUÇÃO AOS CÓDIGOS POLARES

2.1 INFORMAÇÕES PRELIMINARES E NOTAÇÕES

Os canais binários discretos sem memória (B-DMC) são uma importante classe de canais estudados na teoria da informação. Um exemplo importante deste tipo de canal é o canal binário com apagamento (BEC), que será considerado para fins ilustrativos nos próximos capítulos. A ideia principal dos códigos polares é construir, a partir de *N* cópias independentes de um (B-DMC) *W*, um novo conjunto de *N* canais, $W_N^{(i)}$, com 1≤ *i* ≤ *N*, usando uma transformação linear. Quanto mais *N* aumenta, mais esses novos canais $W_N^{(i)}$ são polarizados (o conceito de polarização será descrito na Seção 2.2). Será utilizada neste trabalho a notação $W: X \rightarrow Y$ para denotar um canal binário discreto sem memória (B-DMC) com alfabeto de entrada *X*, alfabeto de saída *Y* e probabilidades de transição *W (y | x)*, $x \in X$, $y \in Y$. Considerando um canal BEC, o alfabeto de entrada *X* será sempre uma entrada binária {0, 1}, enquanto Y e as probabilidades de transição podem ser arbitrárias. Escrevemos W^N para denotar o canal correspondente a *N* usos independentes de *W*; portanto, $W^N : X^N \rightarrow Y^N$ com $W^N (y_1^N \mid x_1^N) = \prod_{i=1}^{N} W (y_i \mid x_i)$. Assim, $y_1^N = (y_1, y_2, ..., y_N)$ serão as observações dos bits de saída, x, do codificador polar, $x_1^N = (x_1, x_2, ..., x_N)$, através de *N* cópias do canal *W,* onde os bits de entrada do codificador polar são $u_1^N = (u_1, u_2, ..., u_N)$.

Dado um B-DMC W, agora é possível definir a informação mútua. Nesta definição X e Y são duas variáveis aleatórias discretas correspondentes a entrada e saída, respectivamente.

Definição 1. A informação mútua de um B-DMC com alfabeto de entrada X = {0, 1} e alfabeto de saída Y arbitrário é definida em [1] como

$$I(W) \triangleq I(X;Y) = \sum_{y \in Y} \sum_{x \in X} \frac{1}{2} W(y|x) \log_2 \frac{W(y|x)}{\frac{1}{2}W(y|0)+\frac{1}{2}W(y|1)} \quad (2.1)$$

em que W (y | x) é a probabilidade de transição do canal para $x \in X$, $y \in Y$.

Definição 2. Para as matrizes A = $\{a_{i,j}\}$ e B = $\{b_{i,\ j}\}$, A ⊗ B representa o Produto de Kronecker, como

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots \\ a_{2,1}B & a_{2,2}B & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \quad (2.2)$$

Definição 3. A matriz NxN de permutação de embaralhamento reverso, R_N, , é definida em [4] como

$$(s_1, s_2, \ldots, s_N) R_N = (s_1, s_3, \ldots, s_{N-1}, s_2, s_4, \ldots, s_N) \quad (2.3)$$

Definição 4. A matriz B_N é a matriz de permutação conhecida como *bit-reversal*, e pode ser definida recursivamente a partir de R_N [4], por

$$B_N = R_N \begin{bmatrix} B_{N/2} & 0 \\ 0 & B_{N/2} \end{bmatrix} = R_N\left(I_2 \otimes B_{N/2}\right) \tag{2.4}$$

em que I_2 é a matriz identidade 2x2 e $B_2 = I_2$.

Definição 5. A permutação bit-reversal é uma ordenação de dados comum, pode ser definida como uma matriz n x n de permutação $P_n x$, para $n = 2^t$ [3], de modo que

$$z = P_n^T x \;\rightarrow\; z(k) = x(r_n(k)) \qquad k=0 \ldots n-1 \tag{2.5}$$

em que $r_n(k)$ é o inteiro obtido pela reversão da ordem de bits na representação de t bits do inteiro k, ou seja

$$(k)_2 = b_0 \ldots b_{t-1} \;\rightarrow\; (r_n(k))_2 = b_{t-1} \ldots b_0 \tag{2.6}$$

Exemplo 1.1: n = 8

Tabela 2.1: Exemplo de permutação bit-reversal

$(k)_{10}$	$(k)_2$	$(r_8(k))_2$	$(r_8(k))_{10}$
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Arikan, em [1], chama B_N de matriz de inversão de bits.

Definição 6. De acordo com [13], um canal gaussiano é um canal discreto com entrada X e saída Y = X + Z, em que Z modela o ruído e é distribuído de acordo com $Z \sim N(0,\sigma^2)$. Para o sinal de entrada binário com ruído gaussiano branco

(AWGN), o alfabeto de entrada é X ∈ {-1, +1}. A função densidade de probabilidade de transição do canal é

$$P_{Y|X}(y|x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-x)^2}{2\sigma^2}}$$

2.2 POLARIZAÇÃO DO CANAL

A polarização do canal é um novo efeito observado por Arikan em [1]. É uma operação pela qual se produz, a partir de N cópias independentes de um determinado B-DMC, W, um segundo conjunto de N canais $W_N^{(i)}$, 1≤ *i* ≤ *N*, de modo a ser possível demonstrar um efeito de polarização. Arikan mostra que para qualquer B-DMC *W*, os canais $W_N^{(i)}$ polarizam-se no sentido de que uma fração dos canais que tenda a ter a informação mútua 1 seja I(W), enquanto a outra fração dos canais que tenda a ter a informação mútua 0 sejam 1 - I (W), quando N ➞ ∞. Esta operação consiste na fase de combinação de canais e na fase de separação de canais.

Na fase de combinação de canais, restringindo-se ao BEC, no qual $x_1^N = c_1^N$, inicialmente se tem cópias de um dado B-DMC, W, de forma recursiva, para produzir um canal de vetor $W_N : X^N \rightarrow Y^N$, em que N = 2^n, n ≥ 0. No nível 0 da recursão (n = 0), temos apenas uma cópia de W, e definimos $W_1 \triangleq W$. O primeiro nível (n = 1) da recursão combina duas cópias do W_1 para obter o canal $W_2 : X^2 \rightarrow Y^2$, conforme mostrado na Figura 2.1 com probabilidades de transição [1]

$$W_2(y_1^2|\, u_1^2) = W\,(y_1|\, u_1 \oplus u_2)\, W\,(y_2|\, u_2) \tag{2.7}$$

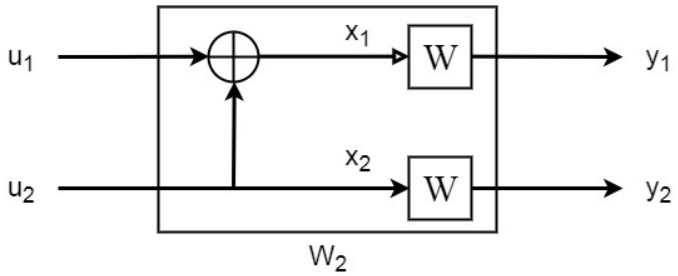


Figura 2.1: O canal W_2

No segundo nível (n = 2) obtemos $W_4 : X^4 \rightarrow Y^4$ de duas cópias de W_2 como mostrado na Figura 2.2. Neste caso, as probabilidades de transição são [1]

$$W_4(y_1^4|\, u_1^4) = W_2\,(y_1^2|\, u_1 \oplus u_2, u_3 \oplus u_4)\, W_2\,(y_3^4|\, u_2, u_4) \tag{2.8}$$

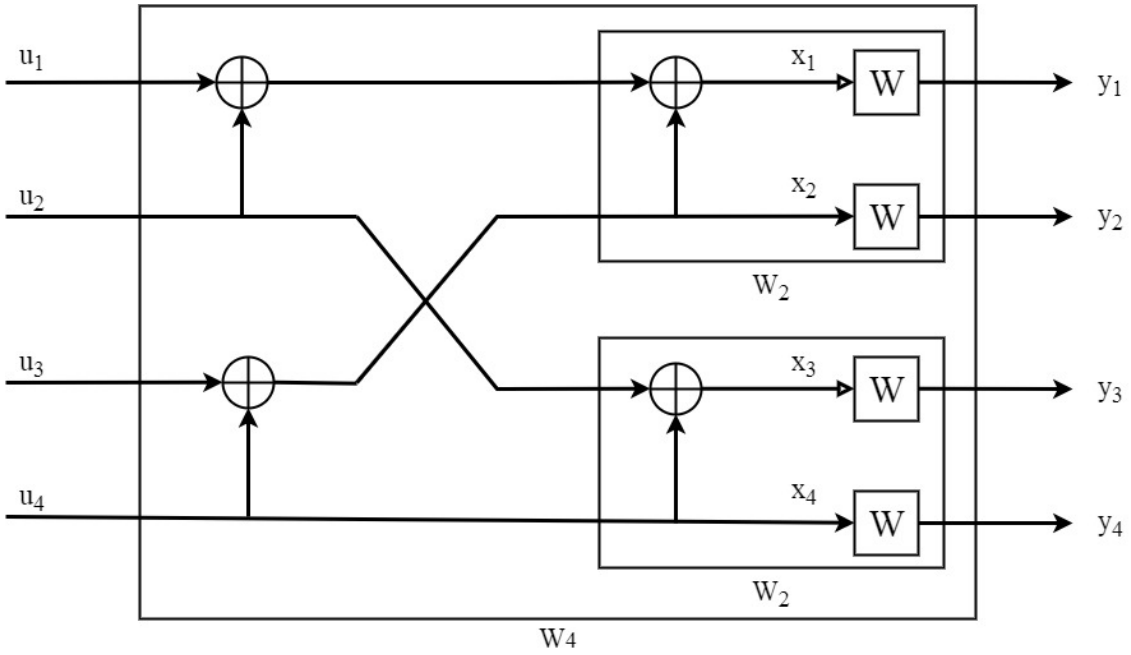


Figura 2.2: O Canal W_4 e suas relações com W_2 e W.

O mapeamento do $u_1^4 \rightarrow x_1^4$ da entrada de W_4 para a entrada de W^4 pode ser escrito como $x_1^4 = u_1^4.G_4$ em que G_N é a matriz geradora. Portanto, obtemos a relação $W_4 (y_1^4 \mid u_1^4) = W^4 (y_1^4 \mid u_1^4 .G_4)$ entre as duas probabilidades de transição [1].

Assim, o mapeamento geral $u_1^N \rightarrow x_1^N$ pode ser escrito por indução e pode ser representado por G_N para que $x_1^N = u_1^N.G_N$ e as probabilidades de transição dos dois canais W_N e W^N são relacionadas por

$$W_N(y_1^N \mid u_1^N) = W^N (y_1^N \mid u_1^N G_N) \tag{2.9}$$

para todo $u_1^N \in X^N$, $y_1^N \in Y^N$ em que $W^N (y_1^N \mid u_1^N .G_N)$ é o canal vetorial que contém a transformação [1].

Na fase de separação de canais, tendo sintetizado um vetor de canais W_N de W^N, se separa W_N de volta em um conjunto de N canais $W_N^{(i)}: X \rightarrow Y^N \times X^{i-1}$, $1 \leq i \leq N$, definido com probabilidade de transição [1]

$$W_N^{(i)}\left(y_1^N, u_1^{i-1} \middle| u_i\right) \triangleq \sum_{u_{i+1}^N \in X^N} \frac{1}{2^{N-1}} W_N(y_1^N \mid u_1^N)$$

em que $\left(y_1^N, u_1^{i-1}\right)$ denotam as saídas de $W_N^{(i)}$ e u_i suas entradas, de modo a ser possível demonstrar um efeito de polarização.

2.3 CODIFICAÇÃO

Um código polar (N,K) é um código de blocos com K bits de entrada e N bits de saída. A transformada polar de tamanho N é definida, de acordo com Arikan, como

$$G_N = B_N \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n} = B_N G_2^{\otimes n} \tag{2.10}$$

em que

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{2.11}$$

G_N é chamado de matriz geradora de tamanho N x N. Sua entrada é um vetor linha u_1^N em que $u_1^N = (u_1, u_2, ..., u_N)$, incluindo os bits congelados e a saída é um vetor linha x_1^N onde $x_1^N = (x_1, x_2, ..., x_N)$. A idéia principal da transformação no codificador polar é criar um conjunto de canais com capacidade C ➞ 1 para N ➞ ∞ e colocar os bits de informação nestes canais, uma vez que são quase livres de ruído. Os restantes N - K bits congelados são transmitidos nos canais ruidosos com C ➞ 0 e são conhecidos no transmissor e no receptor, geralmente fixados em 0. Então a palavra código $x_1^N = u_1^N G_N$ é transmitida pelo canal W e obtemos $y_1^N = x_1^N + z_1^N$, em que z é o ruído introduzido pelo canal. A codificação de 2 bits é mostrada na Figura 2.3 em que, usando a notação apresentada antes, $x_1^2 = u_1^2 G_2$, temos por exemplo $(x_1, x_2) = (u_1, u_2).\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, em que $x_1 = (u_1 \oplus u_2)$ e $x_2 = u_2$.

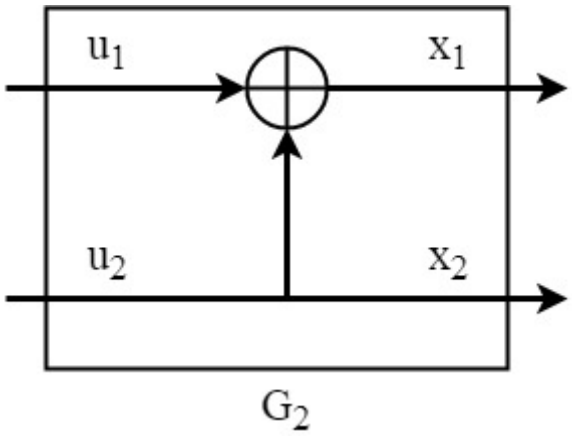


Figura 2.3: Esquemático de G_2

Exemplo 2.1: Para codificação com N = 4 bits é necessário utilizar a matriz G_4, obtendo $x_1^4 = u_1^4 G_4$, em que

$$G_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Neste caso, temos $x_1 = (u_1 \oplus u_2 \oplus u_3 \oplus u_4)$, $x_2 = (u_3 \oplus u_4)$, $x_3 = (u_2 \oplus u_4)$ e $x_4 = u_4$. A Figura 2.4 mostra o bloco principal do nosso esquema de codificação polar. Há dois tipos diferentes de nós: nó de verificação (CN) e nó variável (VN). Esses nós específicos são os mesmos usados para códigos de verificação de paridade de baixa densidade (LDPC). Para o CN o bit de saída é o XOR, ou exclusivo, entre os dois bits de entrada, enquanto que para VN o bit de saída é a repetição do bit de entrada, como podemos ver nos exemplos abaixo.

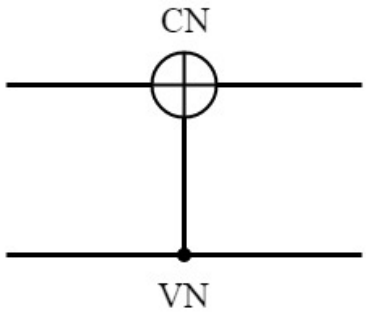


Figura 2.4: Bloco principal dos códigos polares

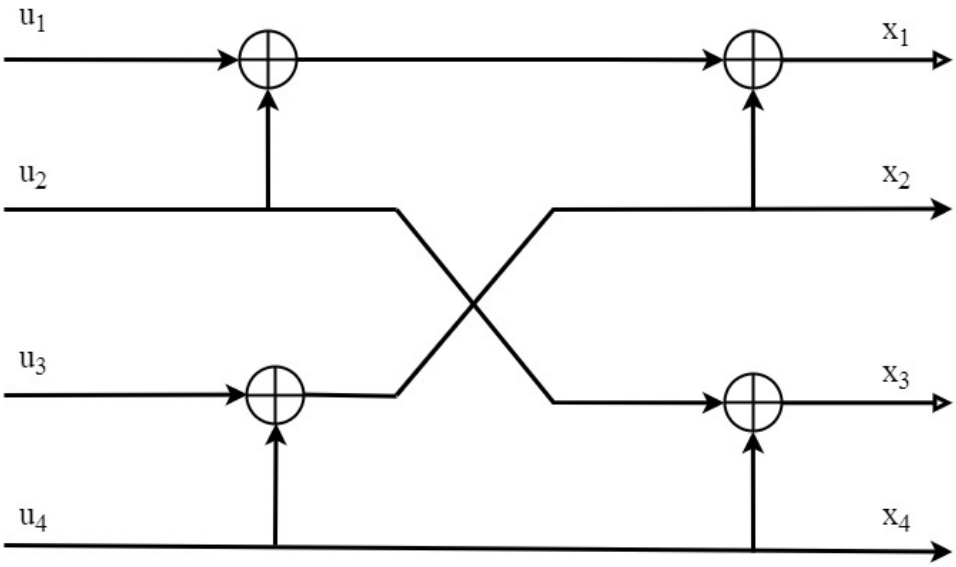


Figura 2.5: Esquema de codificação para N=4

Exemplo 2.2: Para codificação com N = 8 bits é necessário utilizar a matriz G_8, obtendo $x_1^8 = u_1^8 G_8$, em que

$$G_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

A Figura 2.5 e a Figura 2.6 mostram o esquema de codificação para N = 4 e N = 8, respectivamente.

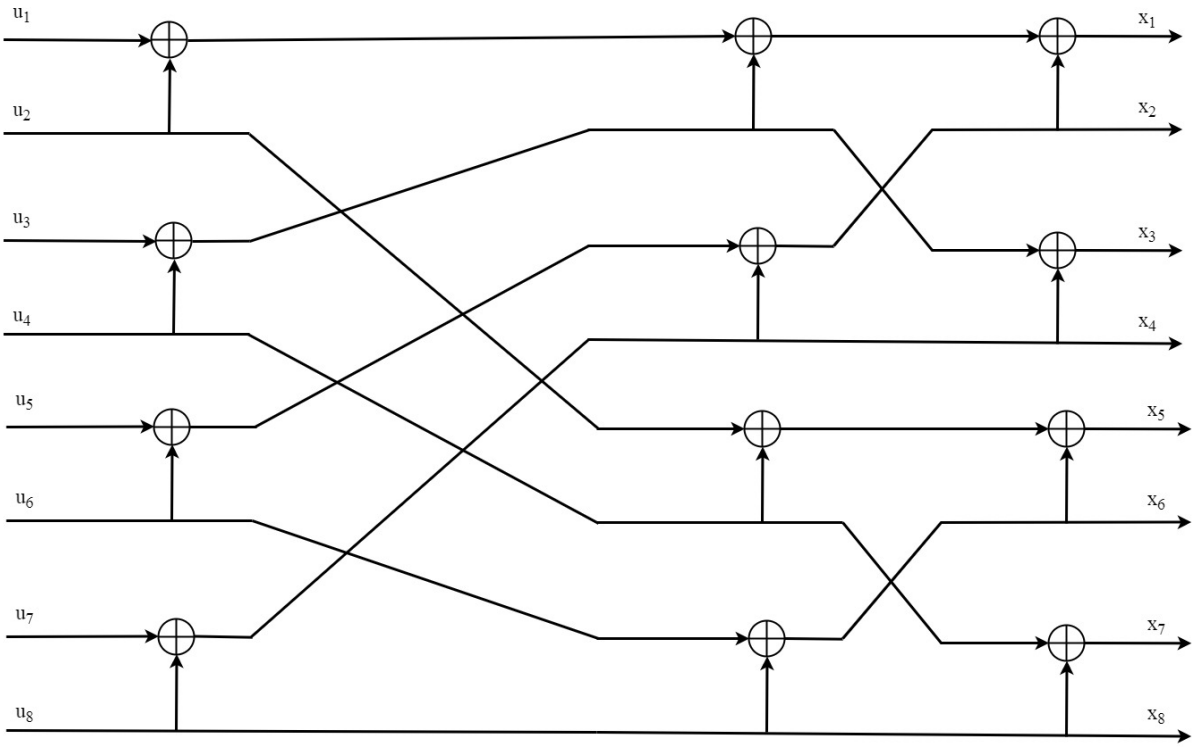


Figura 2.6: Esquema de codificação para N=8

Em [4], é demonstrado várias definições recursivas de G_N que podem ser úteis para diferentes representações. Neste trabalho será apresentado

$$G_N = (I_{N/2} \otimes G_2)R_N (I_2 \otimes G_{N/2}). \quad (2.12)$$

Se usarmos a fórmula recursiva (2.12), finalmente obtemos a Figura 2.7, que é uma generalização da Figura 2.5 e Figura 2.6 para N bits. Isso mostra que $x_1^N=u_1^N G_N$ é equivalente a usar um diagrama com N/2 cópias de G_2 como na Figura 2.3 e duas cópias de $G_{N/2}$. O bloco R_N da Definição 3 é uma matriz de permutação de embaralhamento reverso em que as entradas de número ímpares são copiadas para as entradas do primeiro bloco $G_{N/2}$ enquanto as entradas de número par são copiadas para as entradas do segundo bloco $G_{N/2}$. Seja $X_E(N)$ a complexidade de $u_1^N G_N$. Assumindo a complexidade de G_2 como 1 e a complexidade de R_N como N,

portanto $X_E(N) \leq N/2 + N + 2X_E(N/2)$. A partir de $X_E(2) = 3$, por indução obtemos que $X_E(N) \leq 3/2N\log N$ para todos $N = 2^n$, em que $n \geq 1$. Assim, a complexidade de codificação é O(N logN), conforme em [1].

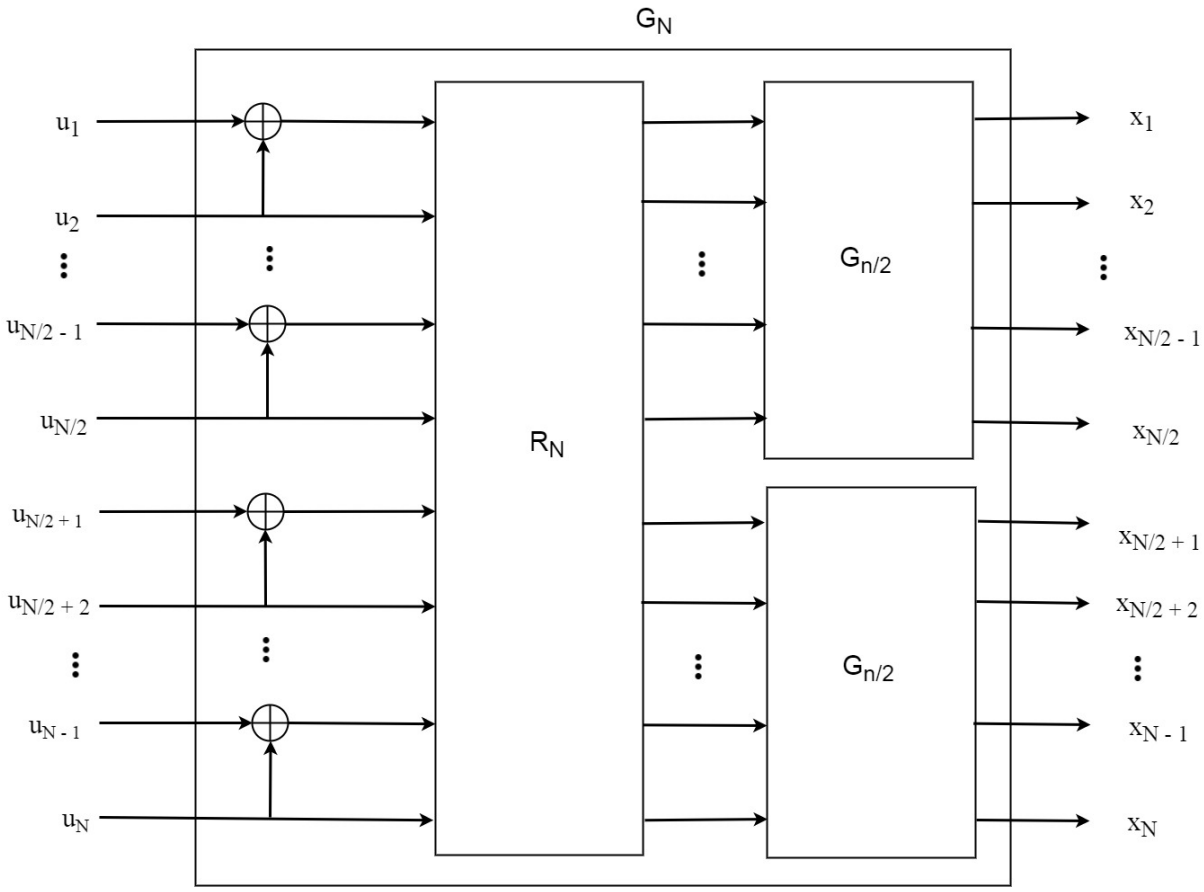


Figura 2.7: Construção recursiva do G_N a partir de duas cópias do $G_{N/2}$

2.4 BEC

O BEC é um canal simétrico ilustrado na Figura 2.8, onde ε é a probabilidade de apagamento. O receptor recebe o bit ou recebe um símbolo "?", que representa que o bit foi apagado. Para canais com simetria entrada-saída, a capacidade é dada por

$$C = max_{p(x)} I(X;Y) \tag{2.13}$$

e para um canal BEC, C = 1 - ε, conforme [1].

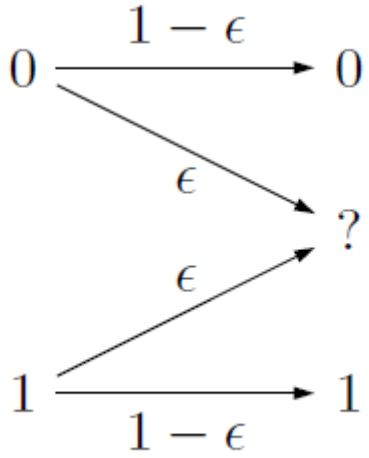


Figura 2.8: BEC. Fonte:[12]

Arikan demonstrou que se W for um BEC, os números $\{I(W_N^{(i)})\}$ poderiam ser computados usando as relações recursivas

$$I(W_N^{(2i-1)}) = I(W_{N/2}^{(i)})^2 \tag{2.14}$$

$$I(W_N^{(2i)}) = 2I(W_N^{(i)}) - I(W_{N/2}^{(i)})^2 \tag{2.15}$$

em que $I(W_1^{(1)}) = 1 - \varepsilon$. Esta transformação significa que dois BEC's independentes são transformados em dois canais polarizados, W^- e W^+, que chamamos de canais "ruim" e "bom", respectivamente. Além disso, uma propriedade importante é a conservação da informação mútua, ou seja, a capacidade de somar dois canais é inalterada, ou seja, $I(W^-) + I(W^+) = 2I(W)$.

Exemplo 2.3: Considerando ε = 0,5. Obtemos C = 1 – ε = 1 - 0,5 = 0,5 = I(W). Uma vez que temos duas fórmulas recursivas, (2.12) e (2.13), para I(W) = 0.5

calculamos $I(W^+) = 2I(W) - I(W)^2 = 0.75$ e $I(W^-) = I(W)^2 = 0,25$, após o primeiro passo W é dividido em um "bom" W^+ e um "ruim" W^- e a soma das capacidades ainda é preservada de fato $I(W^-) + I(W^+) = 2I(W) = 0,25 + 0,75 = 2 \cdot 0,5 = 1$. Para o segundo passo, obtemos $I(W^{--}) = I(W^-)^2 = 0,0625$, $I(W^{-+}) = 2I(W^-) - I(W^-)^2 = 0,4375$ e $I(W^{+-}) = I(W^+)^2 = 0,5625$, $I(W^{++}) = 2I(W^+) - I(W^+)^2 = 0,9375$ e assim por diante para as seguintes etapas. Esse comportamento pode ser visto na Figura 2.9. Ao aplicar recursivamente essa transformação de polarização nos canais resultantes, o resultado é que "os bons melhoram e os maus pioram". Esta evolução da polarização do canal é mostrada na Figura 2.10 para $N = 2^{10}$. Observando a evolução dos canais, descobrimos que as capacidades da maioria dos novos canais tendem a 1, isto é, sem ruído no BEC, ou 0. Da mesma forma, a probabilidade de erro dos canais ruins ou canais bons é de 0 ou 1. Portanto, a principal ideia da codificação polar é colocar os bits de informação sobre esses canais "bons", cuja capacidade tende a ser 1, e os bits congelados, sobre os "maus".

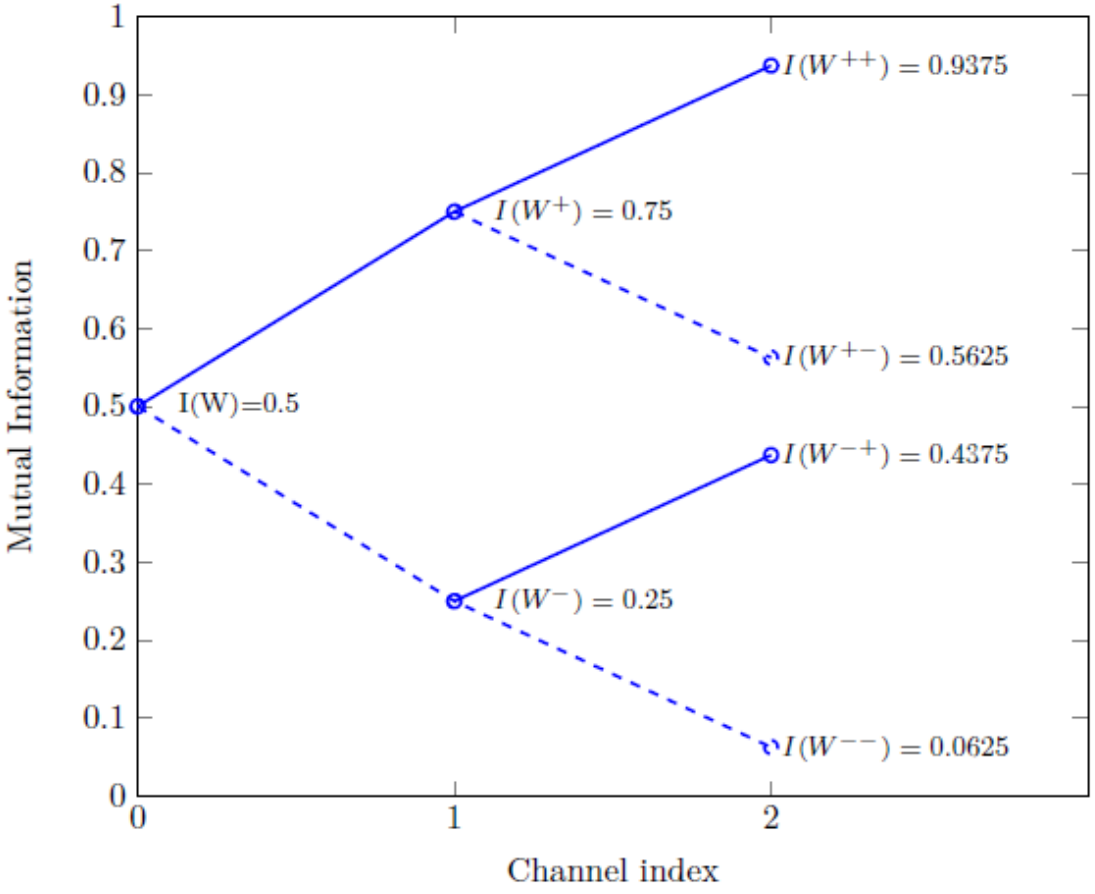


Figura 2.9: Evolução da polarização de canal para N=4. Fonte: [12]

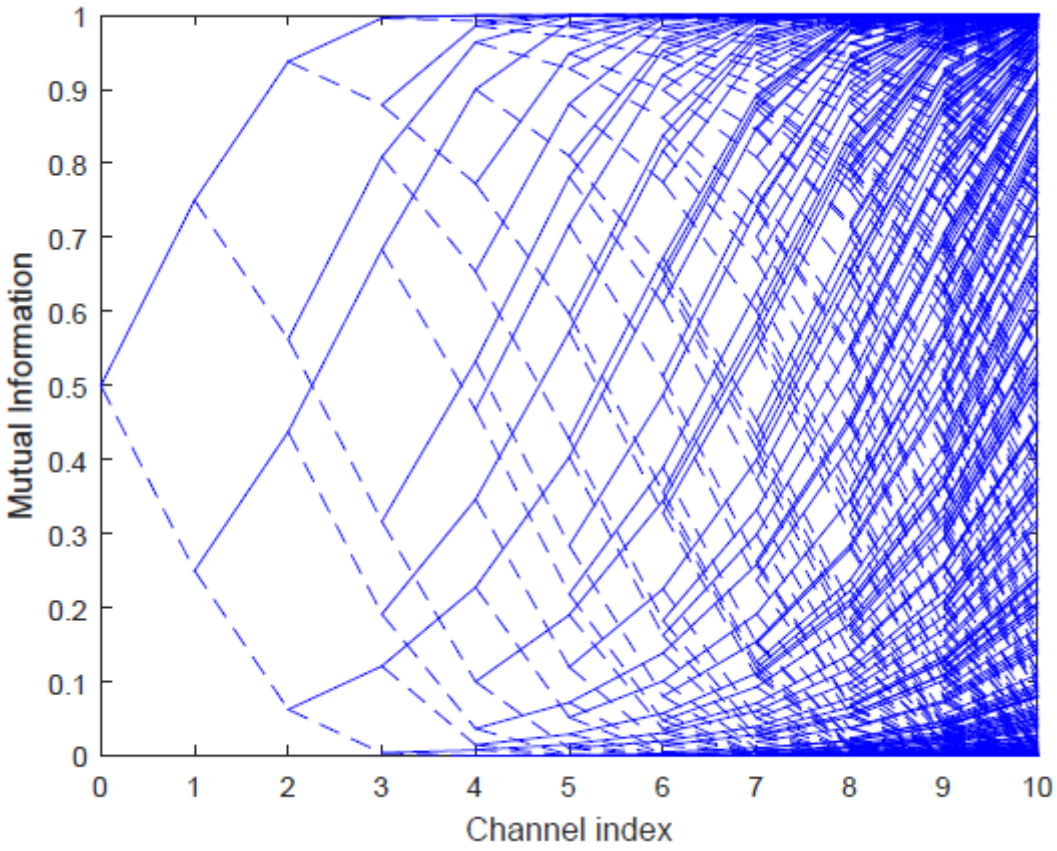


Figura 2.10: Evolução da polarização de canal para N=1024. Fonte:[12]

Exemplo 2.4: Para esclarecer, será descrito o procedimento de polarização de canal e codificação para N = 8, seguindo a Figura 2.12 e 2.13, em que ε, a probabilidade de apagamento, é ½ e R, taxa de código, é ½.

Primeiramente é feita a polarização dos 8 canais, conforme Figura 2.11. A seguir é feito um ranqueamento para saber quais são os canais “bons” e quais são os “ruins”, para que assim seja possível decidir quais serão congelados e quais não serão conforme apresentado na Tabela 2.2. Para os canais “ruins”, é atribuído o valor 0, e para os demais, a informação que deseja-se transmitir.

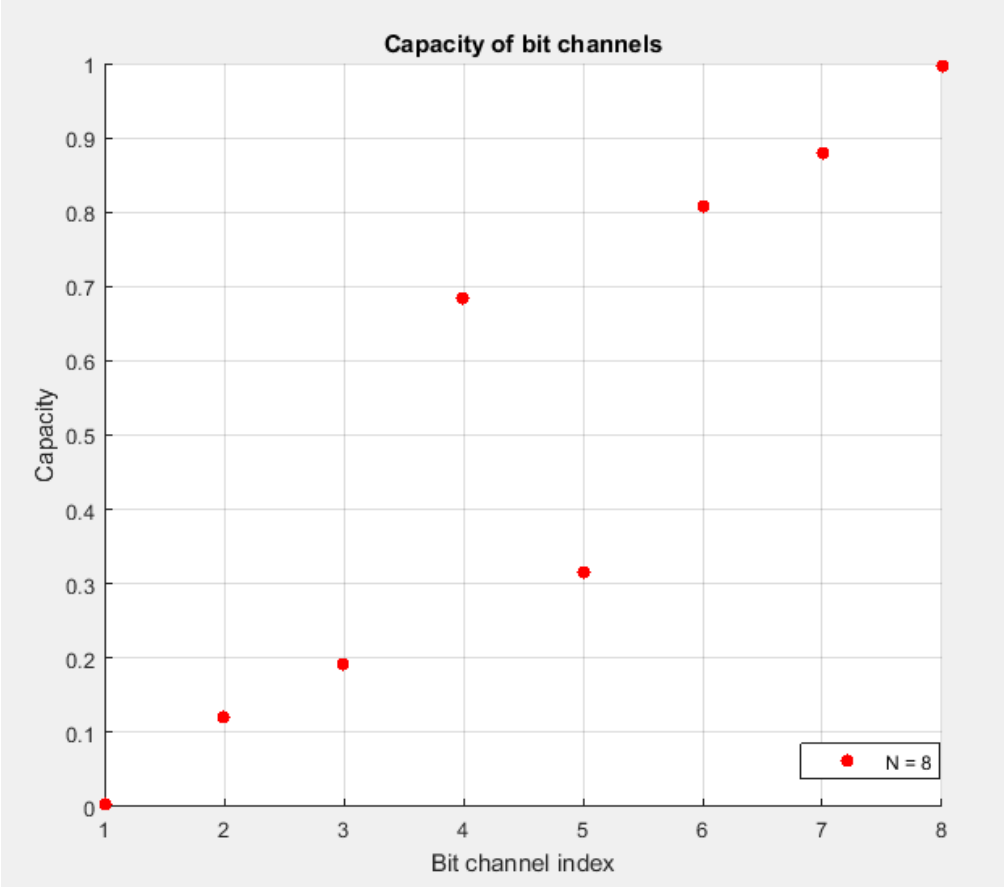


Figura 2.11: Capacidade dos canais para N=8

Tabela 2.2: Dados apresentados no Exemplo 2.4

$I(W_i)$	Rank	Data/frozen	Bit
0.0039	8	frozen	0
0.1211	7	frozen	0
0.1914	6	frozen	0
0.6836	4	data	U_4
0.3164	5	frozen	0
0.8086	3	data	U_6
0.8789	2	data	U_7
0.9961	1	data	U_8

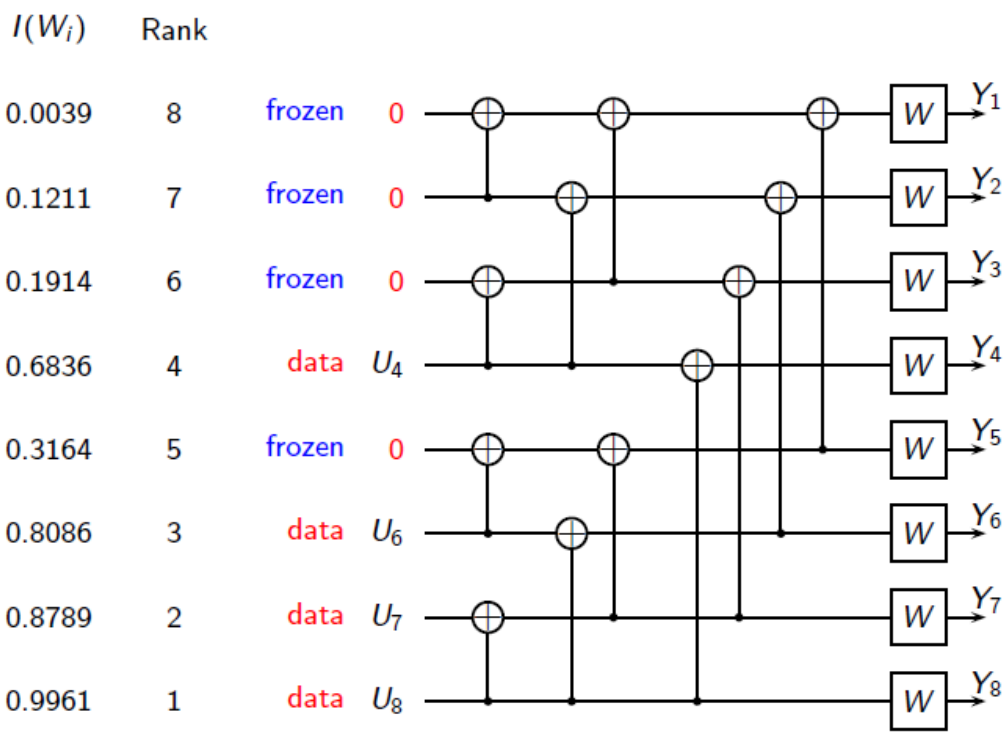


Figura 2.12: Exemplo de codificação polar para W = BEC(½), N=8, R=½.Fonte:Slides de aula Arikan, “Polar Coding Tutorial”.

Considerando que a informação que deseja-se transmitir seja [1101], temos que sua codificação, conforme mostrado na Figura 2.13, é [11000011].

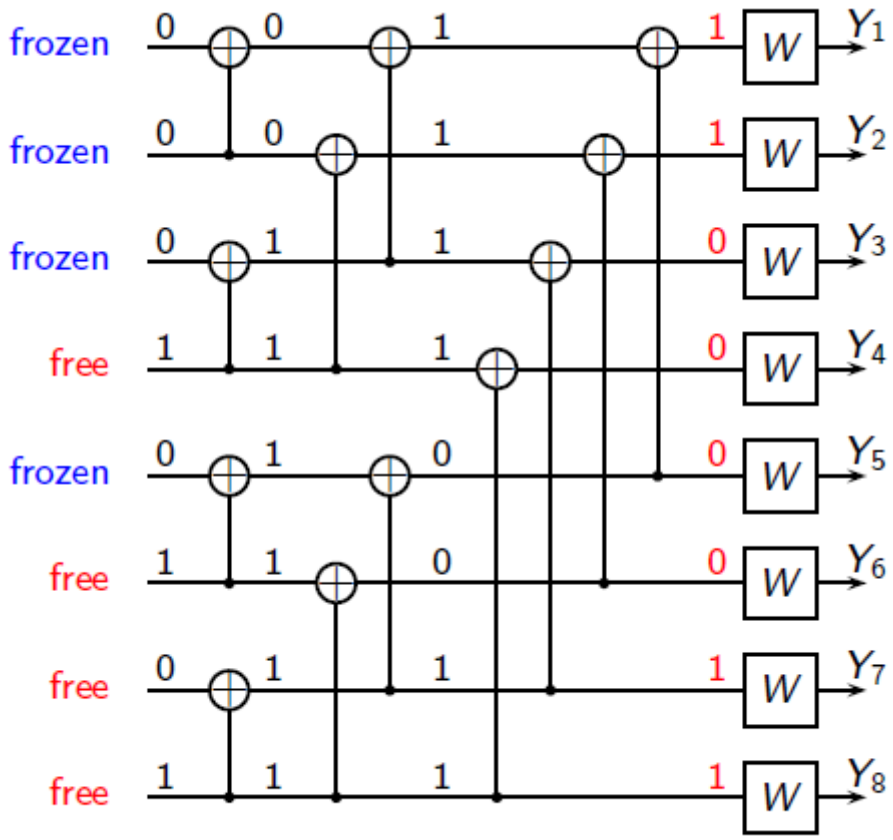


Figura 2.13: Exemplo de codificação polar. Fonte:Slides de aula Arikan, “Polar Coding Tutorial”.

3 DECODIFICAÇÃO

3.1 DECODIFICADOR DE CANCELAMENTO SUCESSIVO

Considere o decodificador de cancelamento sucessivo (SC) recursivo para uma transformada polar de comprimento N, descrito em [1], com $N = 2^n$. O decodificador SC gera uma estimativa $\hat{u}_i$ de u_i observando a saída do canal y_1^N e usando todas as decisões passadas de $\hat{u}_1^{i-1}$.

Se u_i é um bit congelado, o decodificador corrige $\hat{u}_i$ para seu valor conhecido, geralmente 0.

Se u_i for um bit de informação, o decodificador aguarda para estimar todos os bits anteriores e então pode-se calcular um dos três tipos diferentes de métricas, conforme [12] e [13]:

- razão logarítmica de verossimilhança (LLR):

$$\boldsymbol{L_N^{(i)}}\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}}\right) = \mathbf{ln}\left(\frac{\boldsymbol{W_N^{(i)}}\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}} \middle| \boldsymbol{u_i=0}\right)}{\boldsymbol{W_N^{(i)}}\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}} \middle| \boldsymbol{u_i=1}\right)}\right) \tag{3.1}$$

- razão de verossimilhança (LR):

$$LR\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}}\right) = \left(\frac{\boldsymbol{W_N^{(i)}}\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}} \middle| \boldsymbol{u_i=0}\right)}{\boldsymbol{W_N^{(i)}}\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}} \middle| \boldsymbol{u_i=1}\right)}\right) \tag{3.2}$$

- verossimilhança logarítmica (LL):

$$LL\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}}\right) = \left[\ln \boldsymbol{W_N^{(i)}}\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}} \middle| \boldsymbol{u_i = 0}\right), \ln \boldsymbol{W_N^{(i)}}\left(\boldsymbol{y_1^N}, \hat{\mathrm{u}}_1^{\boldsymbol{i-1}} \middle| \boldsymbol{u_i = 1}\right)\right] \tag{3.3}$$

Caso W seja um canal AWGN binário, suas probabilidades de transição são dadas na Definição 6. Portanto, para qualquer saída, o canal LLR correspondente [12]

$$L_i = \ln\left[\frac{P(y_i|0)}{P((y_i|1))}\right] = \ln\left(\frac{\frac{1}{\sigma_n\sqrt{2\pi}} exp\left(-\frac{(y-1)^2}{2\sigma_n^2}\right)}{\frac{1}{\sigma_n\sqrt{2\pi}} exp\left(-\frac{(y+1)^2}{2\sigma_n^2}\right)}\right) = \frac{2}{\sigma_n^2} y_i \tag{3.4}$$

Como y_i é normalmente distribuído com média 1 e variância σ_n^2, então $L_N(i) = 2y_i/\sigma_n^2$ é uma variável normalmente distribuída com média $2/\sigma_n^2$ e variância $4/\sigma_n^2$, conforme [12] e [13].

Em [10], mostrou-se que a decodificação SC pode ser implementada eficientemente conforme a figura 3.1, que mostra o grafo do decodificador SC para n = 8. As razões de verossimilhança do canal (LRs) λ_i são supostamente disponíveis no lado direito do grafo, enquanto os bits estimados, u_i, estão no lado esquerdo. O

decodificador SC é composto de m = $\log_2$ n estágios cada um, contendo n nós. Um nó específico é referido como $N_{l,\,j}$, em que l designa o índice do estágio(0<l<m-1) e j designa o índice do nó no estágio l(0<j<n-1). Cada nó atualiza sua saída de acordo com uma das duas seguintes regras de atualização:

$$\begin{cases} f(a,b) = \frac{1+ab}{a+b} \\ g_{\hat{u}s}(a,b) = a^{1-2\hat{u}s}b \end{cases} \tag{3.2}$$

Os valores a e b são razões de verossimilhança enquanto $\hat{u}_s$ é um bit que representa a soma parcial módulo 2 dos bits previamente estimados. Por exemplo, no nó $N_{1,3}$, apresentado na Figura 3.1, a soma parcial de $\hat{u}_s$ é $\hat{u}_4 \oplus \hat{u}_5$. O valor de $\hat{u}_s$ determina se a função g deve ser uma multiplicação ou uma divisão. Essas regras de atualização, conforme [10], são complexas para implementar em hardware, pois envolvem multiplicações e divisões. Na Seção 3.2, será apresentada uma abordagem, seguida em [10], propondo realizar essas operações no domínio logarítmico e aplicar uma aproximação à função *f*. Por enquanto, consideramos que *f* e *g* sejam caixas pretas até retornarmos a elas na Seção 3.2.

A natureza seqüencial do algoritmo induz uma dependência de dados no seu processamento. Observamos que $N_{1,2}$, representado na Figura 3.1, não pode ser atualizado antes que o bit u_1 seja computado e, logicamente, nem antes de u_0 ser conhecido. Para respeitar a dependência de dados, é necessário definir uma sequência. Arıkan propôs dois sequenciamentos para esse tipo de decodificação. Na sequência da esquerda para a direita, os nós chamam recursivamente seus predecessores até que um nó atualizado seja atingido. A natureza recursiva deste sequenciamento é especialmente adequada, conforme [10] para implementação de software. Na sequência alternativa, da direita para a esquerda, qualquer nó atualiza seu valor sempre que suas entradas estão disponíveis. Cada bit $\hat{u}_i$ é estimado sucessivamente ao ativar a árvore iniciada em $N_{0,\,j}$.

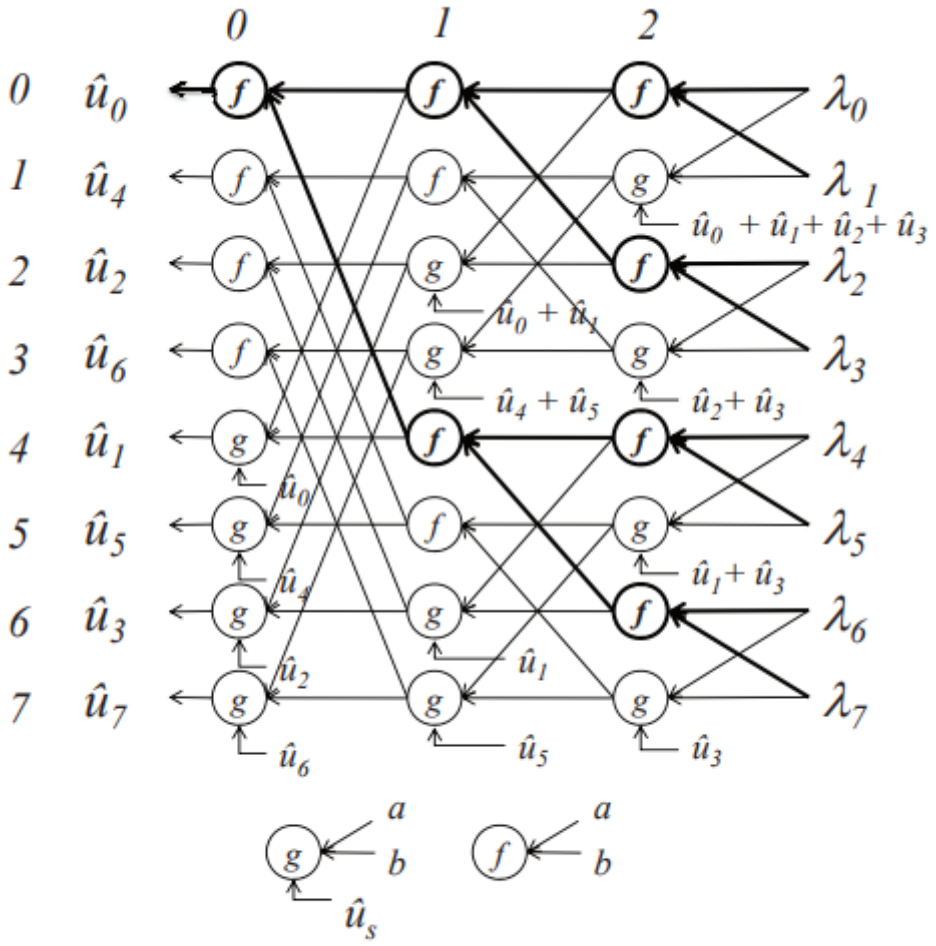


Figura 3.1: Arquitetura de um decodificador SC para n = 8. Fonte:[10]

3.2 DECODIFICADOR SIMPLIFICADO

Em [10], a decodificação SC no domínio do logarítmico foi proposta para reduzir a complexidade dos blocos de computação *f* e *g*. Assumindo que a informação do canal esteja disponível, como as razões logarítmicas de verossimilhança (LLRs) L_i, no domínio LLR, *f* e *g* se tornam

$$\begin{cases} f(L_a, L_b) = 2\tanh^{-1}\left(\tanh\left(\frac{L_a}{2}\right)\tanh\left(\frac{L_b}{2}\right)\right) \\ g_{\hat{u}_s}(L_a, L_b) = L_a(-1)^{\hat{u}_s} + L_b \end{cases} \tag{3.2}$$

em que L_a e L_b são LLRs. Em termos de implementação de hardware, conforme [10], *g* pode ser facilmente implementado com um somador/subtrator controlado pelo bit $\hat{u}_s$. No entanto, *f* envolve algumas funções transcendentais que são complexas de implementar em hardware. Pode-se notar que as funções *f* e *g* são idênticas às regras de atualização usadas na decodificação de propagação de "crenças",BP, de códigos LDPC. Conseqüentemente, semelhante ao que é feito na

implementação do decodificador LDPC [11], *f* pode ser aproximado com a função, apresentada em [10], mínima tal que

$$f(L_a, L_b) \approx \text{sign}(L_a)\text{sign}(L_b)\min(|L_a|, |L_b|) \quad (3.4)$$

Para estimar a degradação do desempenho devido a essa aproximação, os autores em [10] simularam o desempenho de diferentes códigos polares em um canal AWGN com modulação BPSK. Não houve perda significativa de desempenho, conforme [10].

O decodificador SC calcula fase por fase as fórmulas necessárias e decide sequencialmente a estimativa bit a bit. Assim, para i = 1, 2,. . . N, se u_i estiver congelado (frozen), $u_i = 0$, se não gera uma decisão como:

$$\hat{u}_i = \begin{cases} 0, & \text{se } L_N^{(i)} > 0 \\ 1, & \text{caso contrário} \end{cases} \quad (3.5)$$

Para esclarecer, será descrito o procedimento de decodificação para N = 4, seguindo a Figura 3.2.

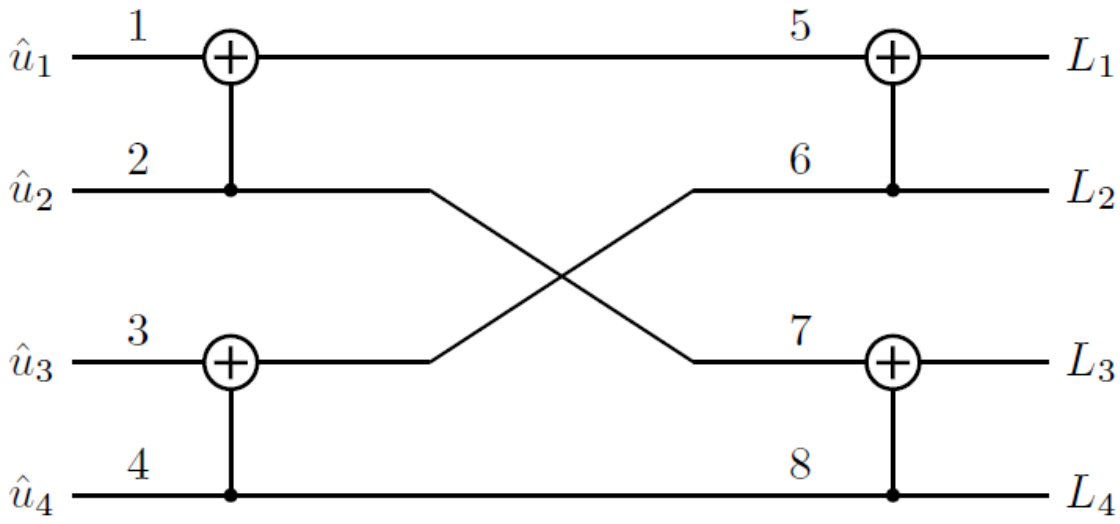


Figura 3.2: Decodificador para N=4

Inicialmente, o decodificador ativa o nó 1 para saber o valor $\hat{u}_1$. Entretanto, para que tal procedimento seja feito, o valor dos nós 5 e 7 devem ser conhecidos, e os nós 5 e 7 precisam dos valores de L_1, L_2, L_3 e L_4 sequencialmente. Para se determinar L_5 e L_7 utiliza-se a função *f*(3.4). Assim temos

$L_5 = f(L_1,L_2) \approx \text{sign}(L_1).\text{sign}(L_2).\min(|L_1|, |L_2|)$
$L_7 = f(L_3,L_4) \approx \text{sign}(L_3).\text{sign}(L_4).\min(|L_3|, |L_4|)$

Assim com os valores nos nós 5 e 7, é possível calcular a LLR no nó 1 como

$L_9 = f(L_5,L_7) \approx sign(L_5).sign(L_7).min(|L_5|, |L_7|)$

Com o valor de L_9 conhecido, é feita a decisão de qual valor, 0 ou 1, conforme 3.5, caso u_1, não seja um bit congelado. Caso seja um bit congelado, $\hat{u}_1$ será 0. Com o valor de $\hat{u}_1$ conhecido é possível calcular o valor da LLR no nó 2, L_{10}, com a função g, conforme (3.2). Assim temos,

$L_{10} = g_0(L_5,L_7) = L_5.(-1)^0 + L_7 = L_5 + L_7$, caso $\hat{u}_1 = 0$, e
$L_{10} = g_1(L_5,L_7) = L_5.(-1)^1 + L_7 = L_7 - L_5$, caso $\hat{u}_1 = 1$.

E assim como foi feito no nó 1, com L_{10} conhecido, é feita a decisão, conforme a equação (3.5), e $\hat{u}_2$ é conhecido. Com os bits nos nós 1 e 2 conhecidos, eles são propagados para os nós 5 e 7, respectivamente, em que o nó 5 terá a soma módulo 2 dos valores dos nós 1 e 2, e o nó 7 terá o mesmo valor de 2. Para estimar o valor de $\hat{u}_3$, é necessário saber saber a LLR dos nós 6 e 8, L_6 e L_8, respectivamente. Para determinar L_6 e L_8, utiliza-se a função *g*, de (3.2), utilizando os bits dos nós 5 e 7, que foram propagados de 1 e 2. Assim temos

$L_6 = g_0(L_1,L_2) = L_1.(-1)^0 + L_2 = L_1 + L_2$, caso $\hat{u}_1 \oplus \hat{u}_2 = 0$, e
$L_6 = g_1(L_1,L_2) = L_1.(-1)^1 + L_2 = L_2 - L_1$, caso $\hat{u}_1 \oplus \hat{u}_2 = 1$.

e

$L_8 = g_0(L_3,L_4) = L_3.(-1)^0 + L_4 = L_3 + L_4$, caso $\hat{u}_2 = 0$, e
$L_8 = g_1(L_3,L_4) = L_3.(-1)^1 + L_4 = L_4 - L_3$, caso $\hat{u}_2 = 1$.

Com os valores de L_6 e L_8, é possível encontrar a LLR do nó 3, L_{11}, utilizando a função *f*, em (3.4). Assim temos

$L_{11} = f(L_6,L_8) \approx sign(L_6).sign(L_8).min(|L_6|, |L_8|)$

E como feito anteriormente, utilizando (3.5) é possível tomar a decisão do valor de $\hat{u}_3$. Por fim, para determinar $\hat{u}_4$, utiliza-se a função *g*, em (3.2). Assim temos

$L_{12} = g_0(L_6,L_8) = L_6.(-1)^0 + L_8 = L_6 + L_8$, caso $\hat{u}_3 = 0$, e
$L_{12} = g_1(L_6,L_8) = L_6.(-1)^1 + L_8 = L_8 - L_6$, caso $\hat{u}_3 = 1$.

Então, $\hat{u}_4$ é estimado, conforme (3.5), e o processo de decodificação está completo.

Uma vez que o algoritmo SC segue o mesmo diagrama do codificador, então conforme [1], a complexidade do decodificador SC é a mesma do codificador, ou seja, O(N logN).

4 SIMULAÇÕES

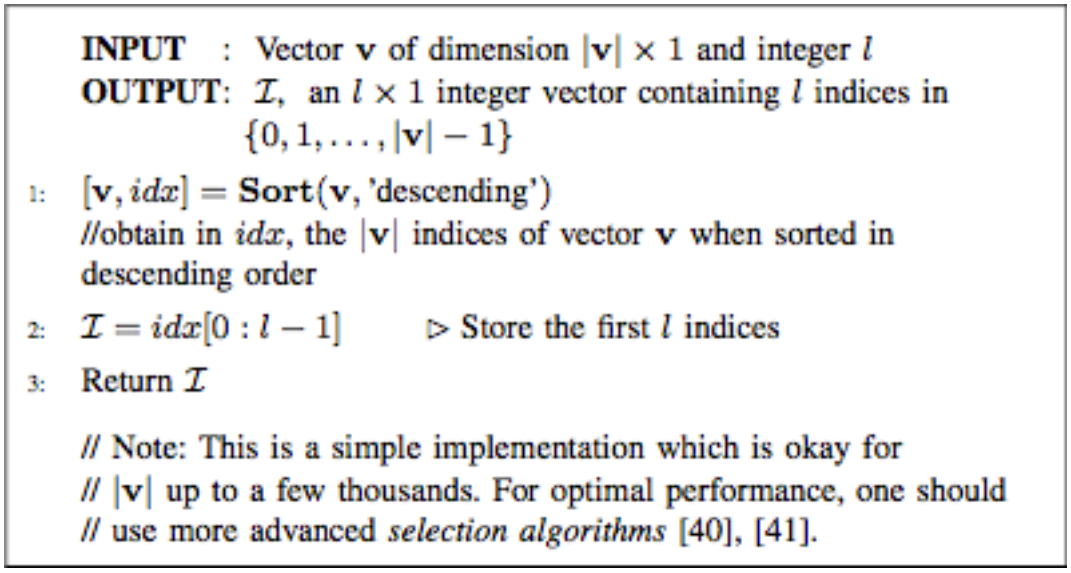
As implementações foram baseadas nos algoritmos apresentados em [15][16][17], e seus pseudo códigos serão apresentado nas figuras 4.1, 4.2, 4.3 e 4.4.

Para a polarização dos canais, [15] representou as equações 2.14 e 2.15, como {z, z} → {2z - z^2, z^2}. Devido à sua simplicidade, esta construção é amplamente utilizada e produziu bons códigos polares [15].

INPUT : N, K, and design-SNR $EdB = (RE_b/N_0$ in dB)
OUTPUT: $\mathcal{F} \subset \{0, 1, \ldots, N-1\}$ with $|\mathcal{F}| = N - K$

1: $S = 10^{EdB/10}$ and $n = \log_2 N$
2: $\mathbf{z}^{(0)} \in \mathbb{R}^N$, initialize $\mathbf{z}^{(0)}[0] = \exp(-S)$
3: **for** $j = 1 : n$ **do** ▷ for each stage in Fig. 1, right-to-left
4: $u = 2^j$
5: **for** $t = 0 : \frac{u}{2} - 1$ **do** ▷ For each connection
6: $T = \mathbf{z}^{(0)}[t]$
7: $\mathbf{z}^{(0)}[t] = 2T - T^2$ ▷ Upper channel
8: $\mathbf{z}^{(0)}[u/2 + t] = T^2$ ▷ Lower channel
9: **end**
10: **end**
11: $\mathcal{F} = \mathbf{indices\_of\_greatest\_elements}\left(\mathbf{z}^{(0)}, N - K\right)$
// Find indices of the greatest $N - K$ elements
12: Return $\mathcal{F}$

Figura 4.1: Pseudo código para polarização dos canais. Fonte: [15]



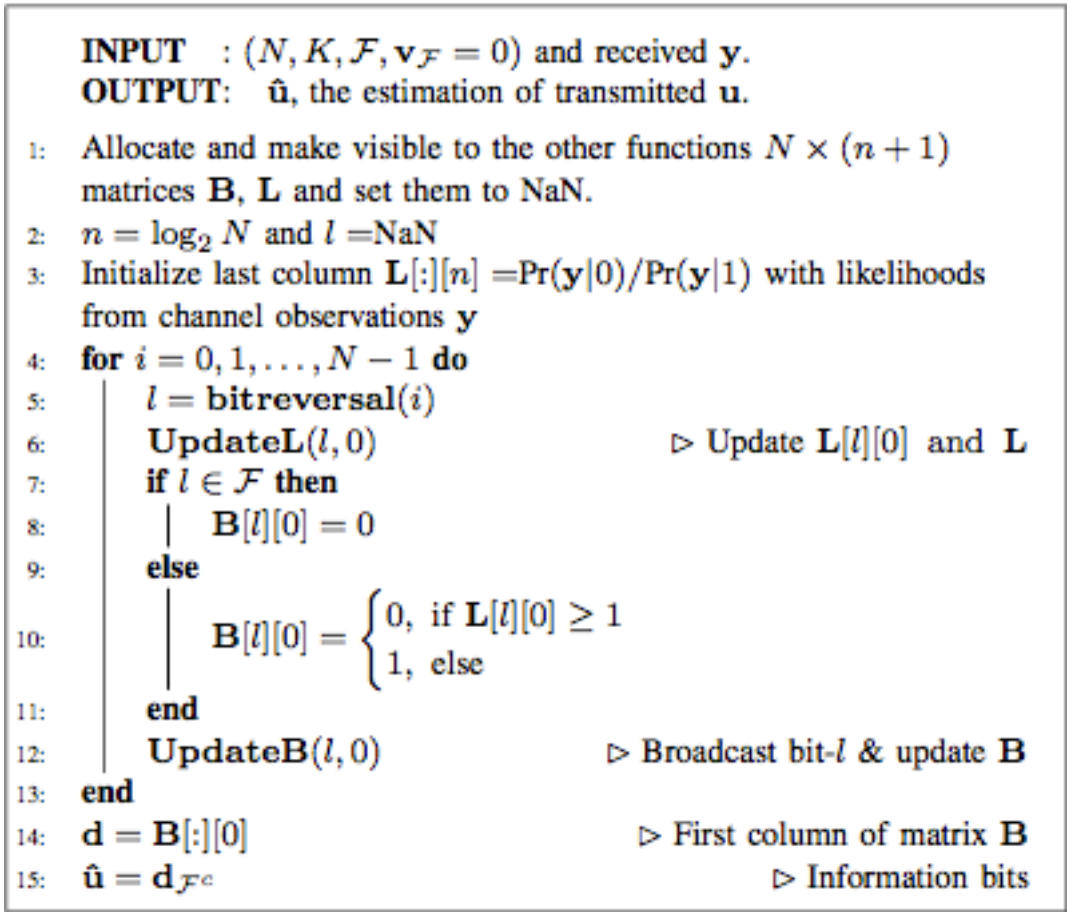
INPUT : Vector $\mathbf{v}$ of dimension $|\mathbf{v}| \times 1$ and integer l
OUTPUT: $\mathcal{I}$, an $l \times 1$ integer vector containing l indices in $\{0, 1, \ldots, |\mathbf{v}| - 1\}$

1: $[\mathbf{v}, idx] = \mathbf{Sort}(\mathbf{v}, \text{'descending'})$
//obtain in idx, the $|\mathbf{v}|$ indices of vector $\mathbf{v}$ when sorted in descending order
2: $\mathcal{I} = idx[0 : l - 1]$ ▷ Store the first l indices
3: Return $\mathcal{I}$

// Note: This is a simple implementation which is okay for
// $|\mathbf{v}|$ up to a few thousands. For optimal performance, one should
// use more advanced *selection algorithms* [40], [41].

Figura 4.2: Pseudo código para a função indices_of_gratest_elements. Fonte: [15]

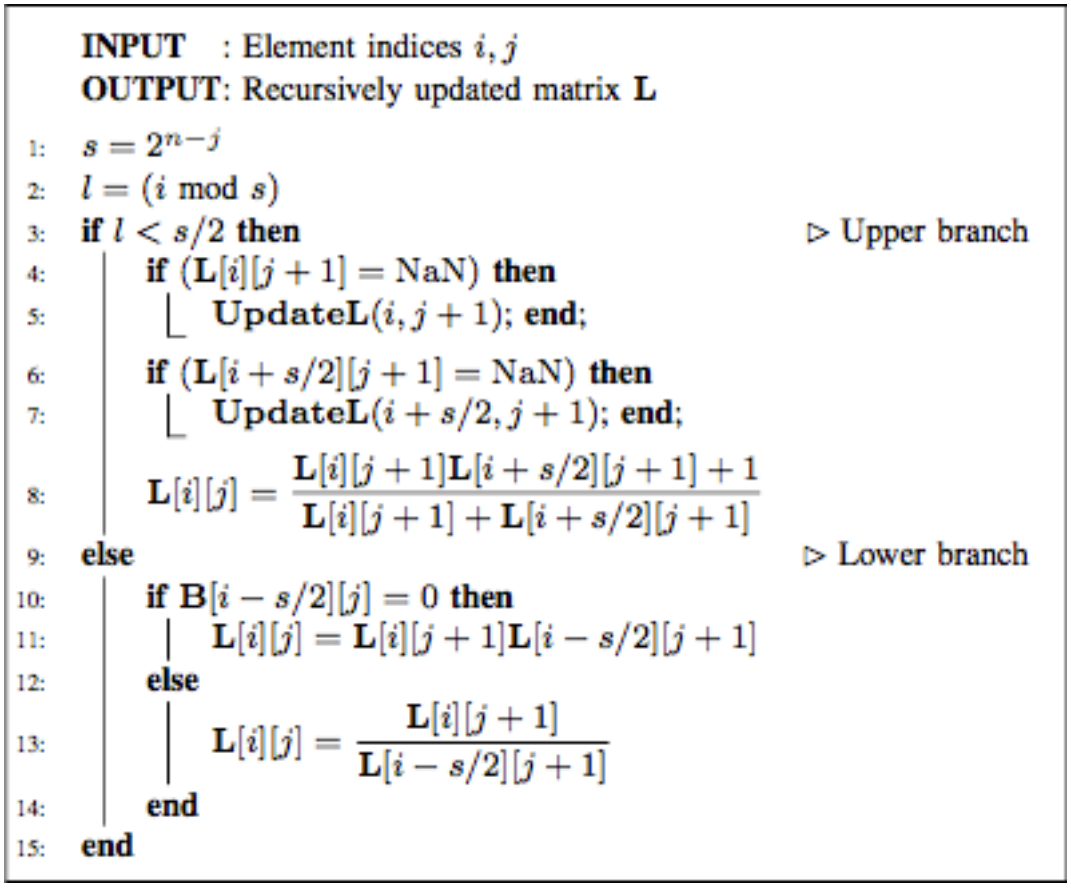
No pseudo código de decodificação por cancelamento sucessivo a matriz B é a matriz dos bits e L é a matriz de razões de verossimilhança.



INPUT : $(N, K, \mathcal{F}, \mathbf{v}_{\mathcal{F}} = 0)$ and received $\mathbf{y}$.
OUTPUT: $\hat{\mathbf{u}}$, the estimation of transmitted $\mathbf{u}$.

1: Allocate and make visible to the other functions $N \times (n + 1)$ matrices $\mathbf{B}$, $\mathbf{L}$ and set them to NaN.
2: $n = \log_2 N$ and l =NaN
3: Initialize last column $\mathbf{L}[:][n]$ =$\Pr(\mathbf{y}|0)/\Pr(\mathbf{y}|1)$ with likelihoods from channel observations $\mathbf{y}$
4: **for** $i = 0, 1, \ldots, N - 1$ **do**
5: $l = \mathbf{bitreversal}(i)$
6: $\mathbf{UpdateL}(l, 0)$ ▷ Update $\mathbf{L}[l][0]$ and $\mathbf{L}$
7: **if** $l \in \mathcal{F}$ **then**
8: $\mathbf{B}[l][0] = 0$
9: **else**
10: $\mathbf{B}[l][0] = \begin{cases} 0, & \text{if } \mathbf{L}[l][0] \geq 1 \\ 1, & \text{else} \end{cases}$
11: **end**
12: $\mathbf{UpdateB}(l, 0)$ ▷ Broadcast bit-l & update $\mathbf{B}$
13: **end**
14: $\mathbf{d} = \mathbf{B}[:][0]$ ▷ First column of matrix $\mathbf{B}$
15: $\hat{\mathbf{u}} = \mathbf{d}_{\mathcal{F}^c}$ ▷ Information bits

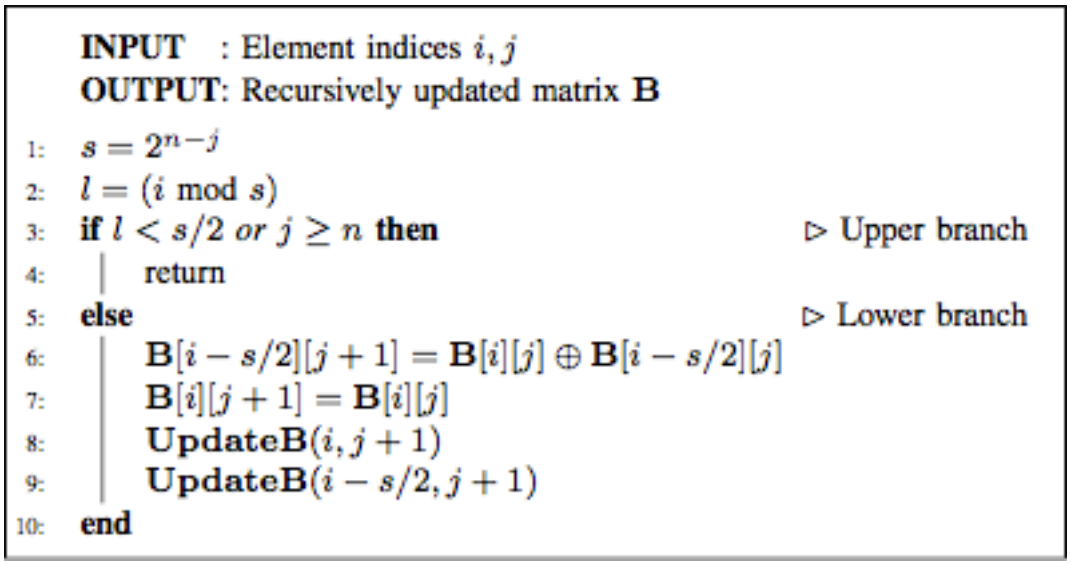
Figura 4.3: Módulo principal para o algoritmo de decodificação por cancelamento sucessivo. Fonte: [17]



INPUT : Element indices i, j
OUTPUT: Recursively updated matrix $\mathbf{L}$

1: $s = 2^{n-j}$
2: $l = (i \bmod s)$
3: **if** $l < s/2$ **then** ▷ Upper branch
4: **if** $(\mathbf{L}[i][j+1] = \text{NaN})$ **then**
5: $\mathbf{UpdateL}(i, j+1)$; **end**;
6: **if** $(\mathbf{L}[i+s/2][j+1] = \text{NaN})$ **then**
7: $\mathbf{UpdateL}(i+s/2, j+1)$; **end**;
8: $\mathbf{L}[i][j] = \dfrac{\mathbf{L}[i][j+1]\mathbf{L}[i+s/2][j+1]+1}{\mathbf{L}[i][j+1]+\mathbf{L}[i+s/2][j+1]}$
9: **else** ▷ Lower branch
10: **if** $\mathbf{B}[i-s/2][j] = 0$ **then**
11: $\mathbf{L}[i][j] = \mathbf{L}[i][j+1]\mathbf{L}[i-s/2][j+1]$
12: **else**
13: $\mathbf{L}[i][j] = \dfrac{\mathbf{L}[i][j+1]}{\mathbf{L}[i-s/2][j+1]}$
14: **end**
15: **end**

Figura 4.4: Pseudo código para a função UpdateL. Fonte: [17]



INPUT : Element indices i, j
OUTPUT: Recursively updated matrix $\mathbf{B}$

1: $s = 2^{n-j}$
2: $l = (i \bmod s)$
3: **if** $l < s/2$ *or* $j \geq n$ **then** ▷ Upper branch
4: return
5: **else** ▷ Lower branch
6: $\mathbf{B}[i-s/2][j+1] = \mathbf{B}[i][j] \oplus \mathbf{B}[i-s/2][j]$
7: $\mathbf{B}[i][j+1] = \mathbf{B}[i][j]$
8: $\mathbf{UpdateB}(i, j+1)$
9: $\mathbf{UpdateB}(i-s/2, j+1)$
10: **end**

Figura 4.5: Pseudo código para a função UpdateB. Fonte: [17]

Nas simulações, comparamos nossos resultados de BER com referências da literatura, para validar as implementações. Para determinar os desempenhos, sempre será utilizada a aproximação de Monte Carlo. A Figura 4.6 mostra uma comparação entre o resultado da decodificação SC de [15] e a simulação realizada para um código polar de comprimento N=2048, taxa R=½ e projetado para

Eb/N0=2dB. Além disso, encontraram-se outros resultados interessantes da literatura com diferentes valores de razão sinal ruído,SNR, projetada. Em particular, [15] mostra o desempenho de um código de comprimento N = 2048, taxa R=½, com valores de SNR projetada variados, que foi aqui simulado e o resultado é apresentado na Figura 4.7. Claramente a SNR projetada é um fator critico nos códigos polares, e a qual é uma função de todos os possíveis parametros como taxa e algoritmo de construção. Esta comparação foi feita por [15] e é apresentada na Figura 4.8. Podemos notar que se podemos achar a otima SNR projetada, qualquer construção produz um código polar com mesmo desempenho.A Figura 4.9 e a Figura 4.10 mostram os desempenhos aqui obtidos de um código polar sob decodificação SC com comprimentos variáveis, taxa R=½ e projetado para SNR=0dB e SNR=2dB, respectivamente, sendo possível perceber que com o aumento de N, o desempenho melhora. Comparando-se com os resultados de [13], é possível observar que os resultados obtidos neste trabalho são coerentes. Finalmente, será apresentada na Figura 4.11 a comparação entre um código LDPC utilizado no padrão WiMAX de comprimento N=2304 e taxa R=½, cujos resultados de simulação foram retirados de [14], e um código polar de comprimento N=2048 e taxa R=½ e projetado para Eb/N0=2dB. O código foi construído usando os métodos de [15], com otimização para SNR=2dB. Da Figura 4.11 podemos notar que o desempenho da decodificação apenar utilizando o cancelamento sucessivo é inferior ao dos códigos LDPC.

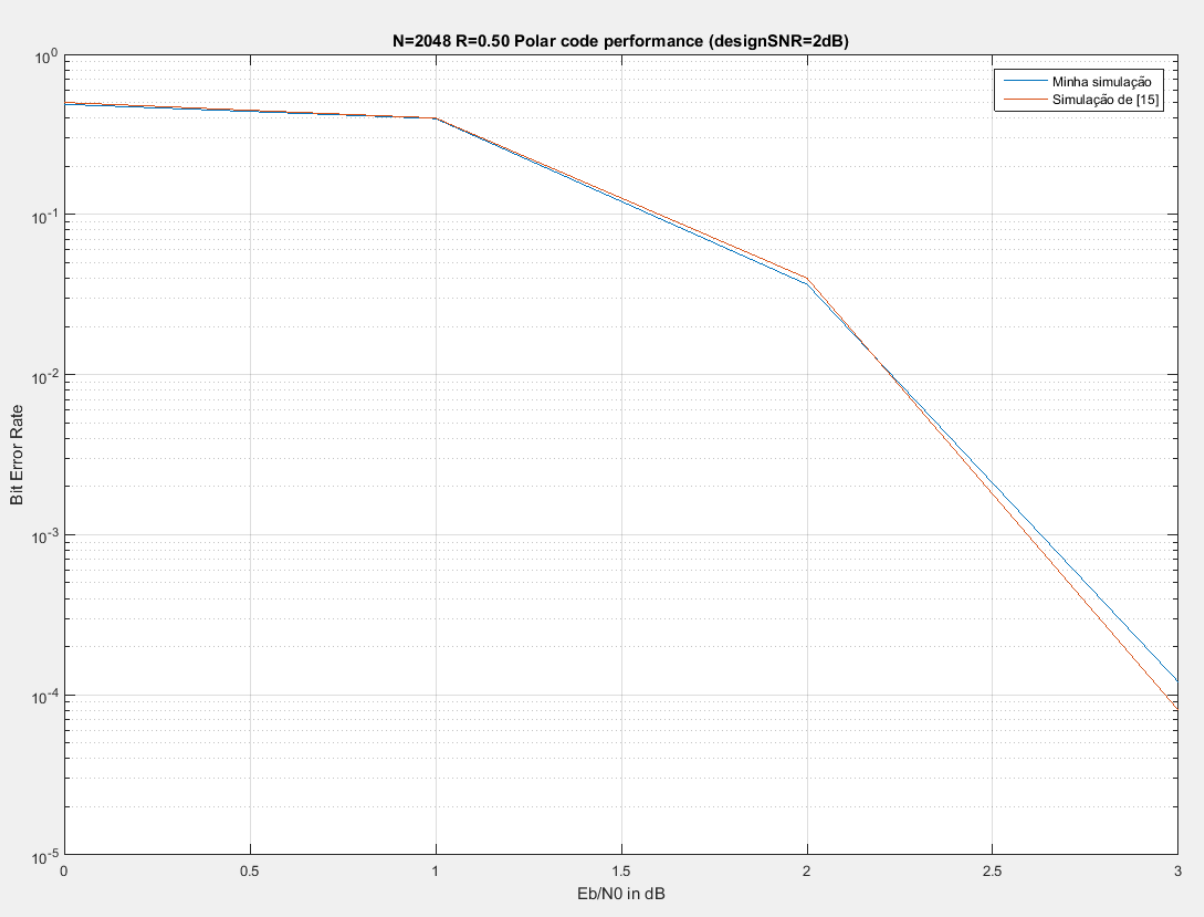


Figura 4.6: Comparação entre simulação realizada neste trabalho e os resultados de [15], onde o comprimento do código polar é N=2048, taxa R=½ e projetado para Eb/N0=2dB.

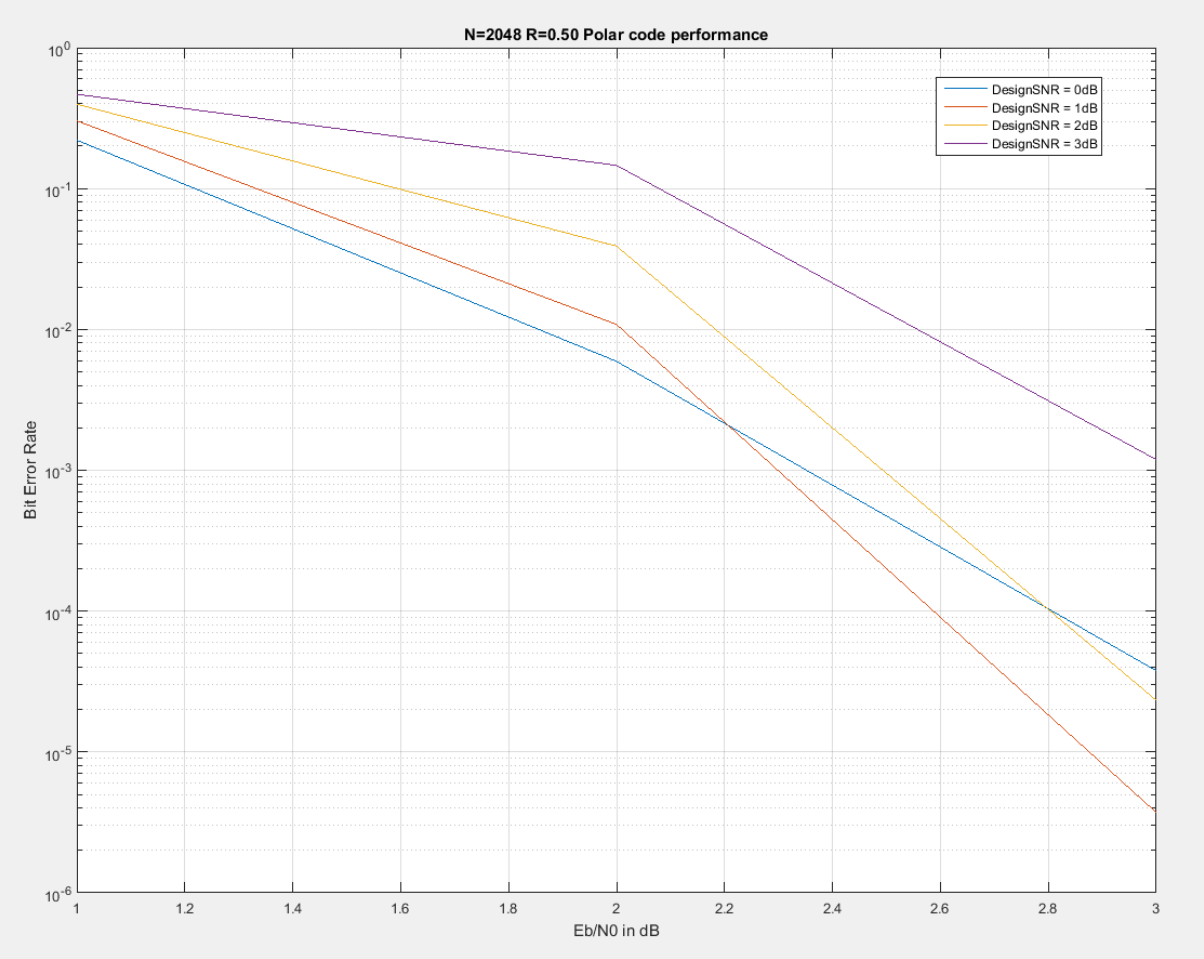


Figura 4.7: Desempenho de código de comprimento N=2048, taxa R=½, e com valores de SNR projetada variados.

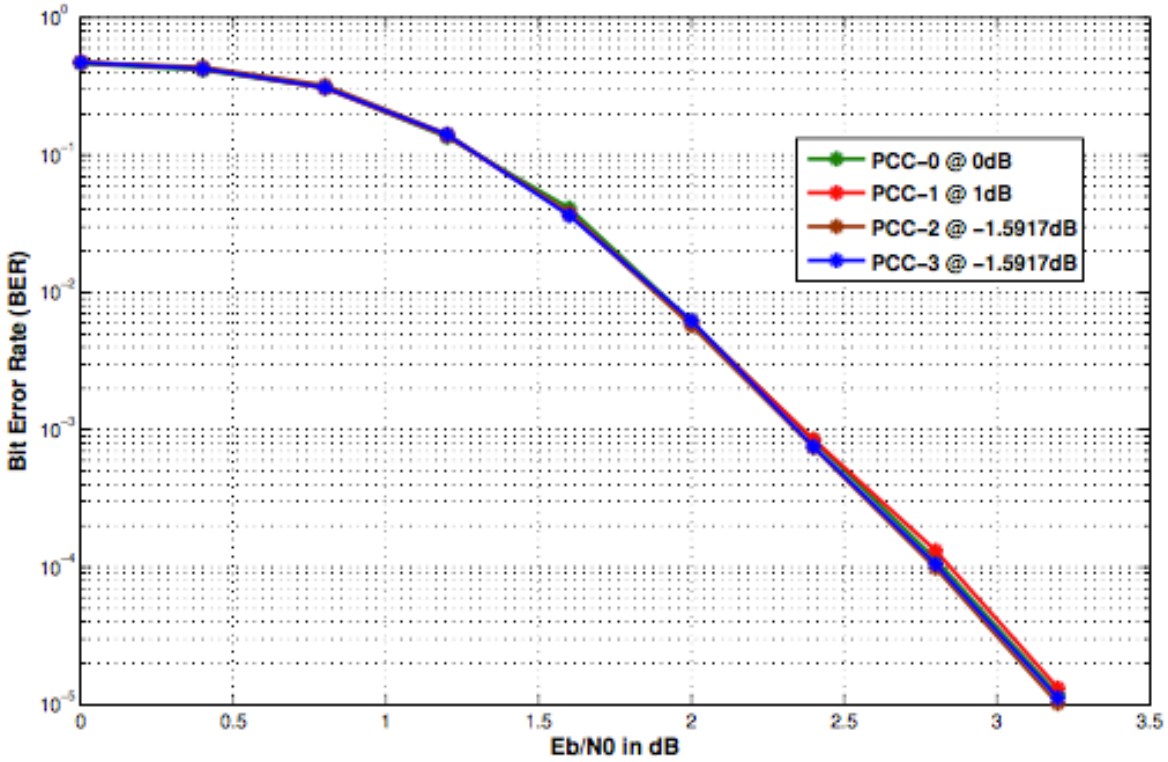


Figura 4.8: Comparação de diferentes construções de PC e suas ótimas SNR projetadas para N=2048 e taxa de ½. Fonte: [15]

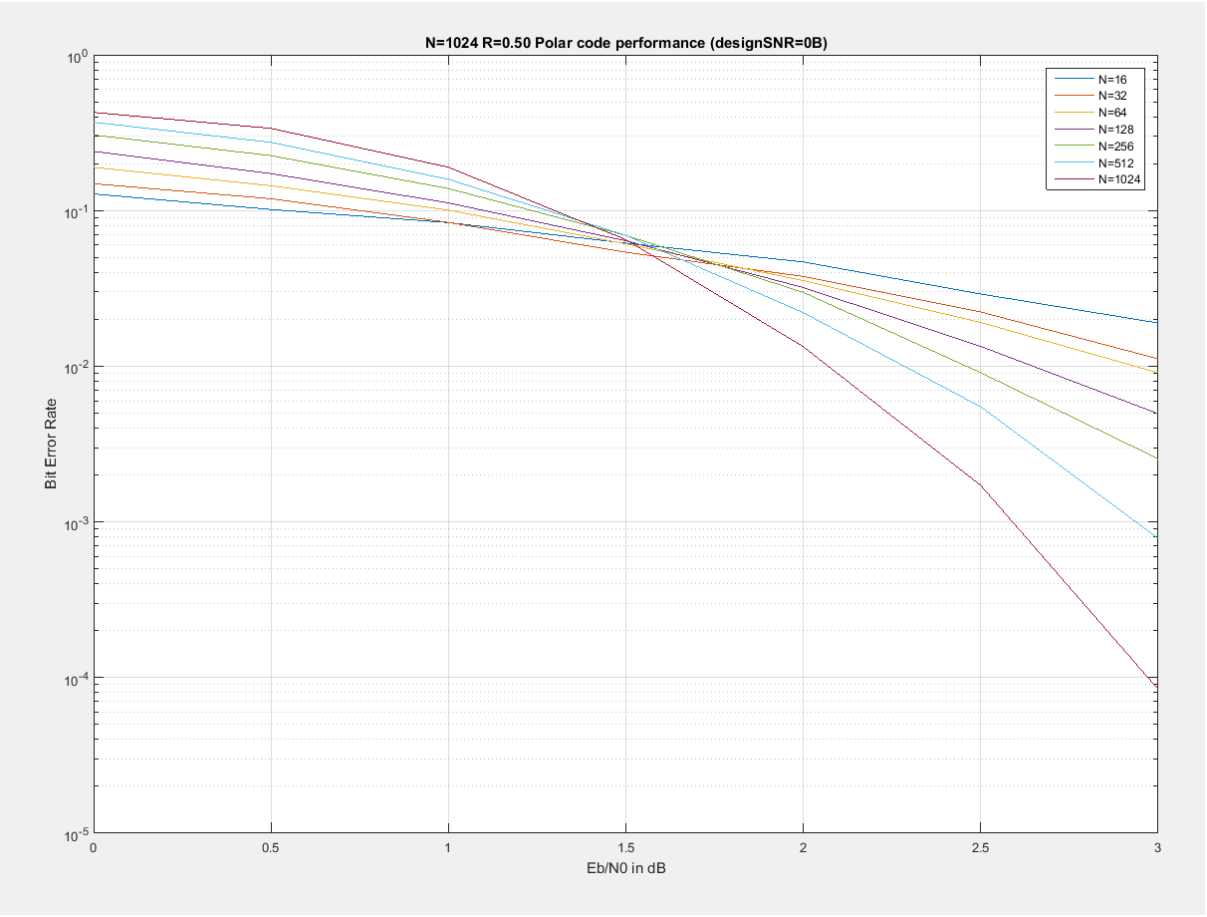


Figura 4.9: Desempenho de código de comprimento variável, taxa R=½ e SNR projetada de 0dB.

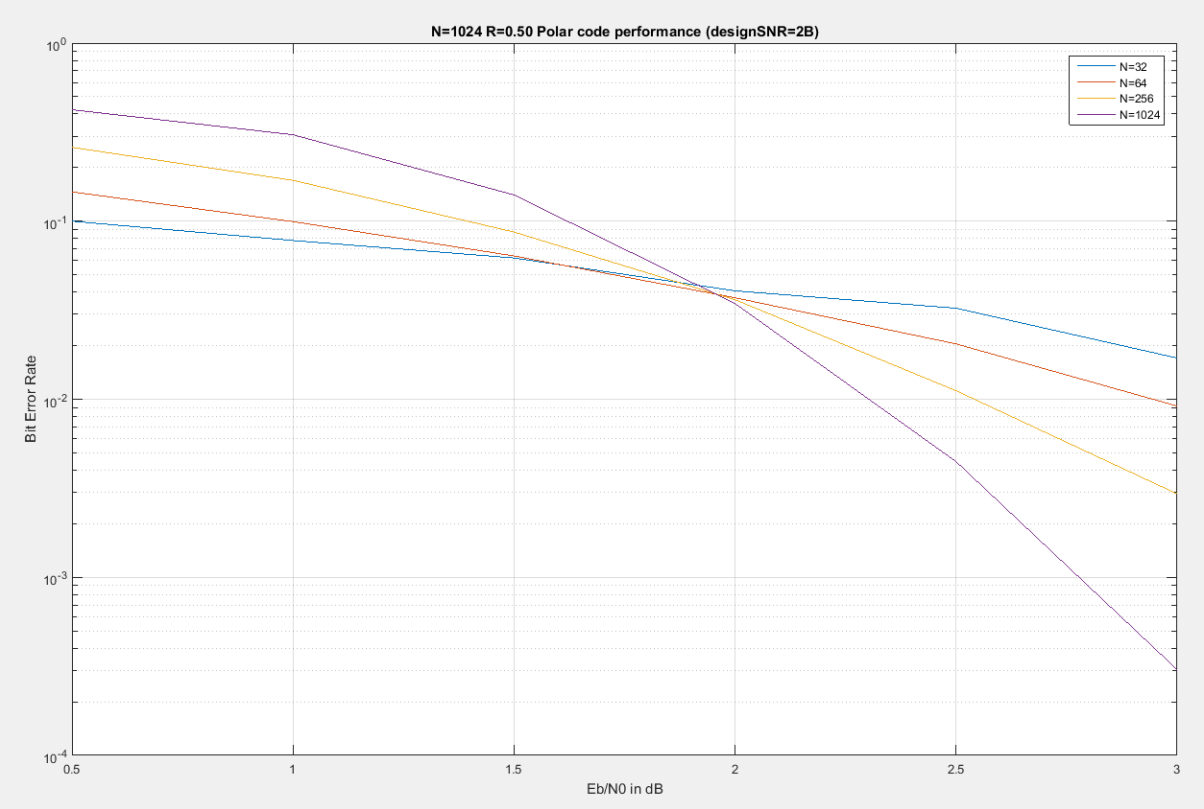


Figura 4.10: Desempenho de código de comprimento variável, taxa R=½ e SNR projetada de 2dB.

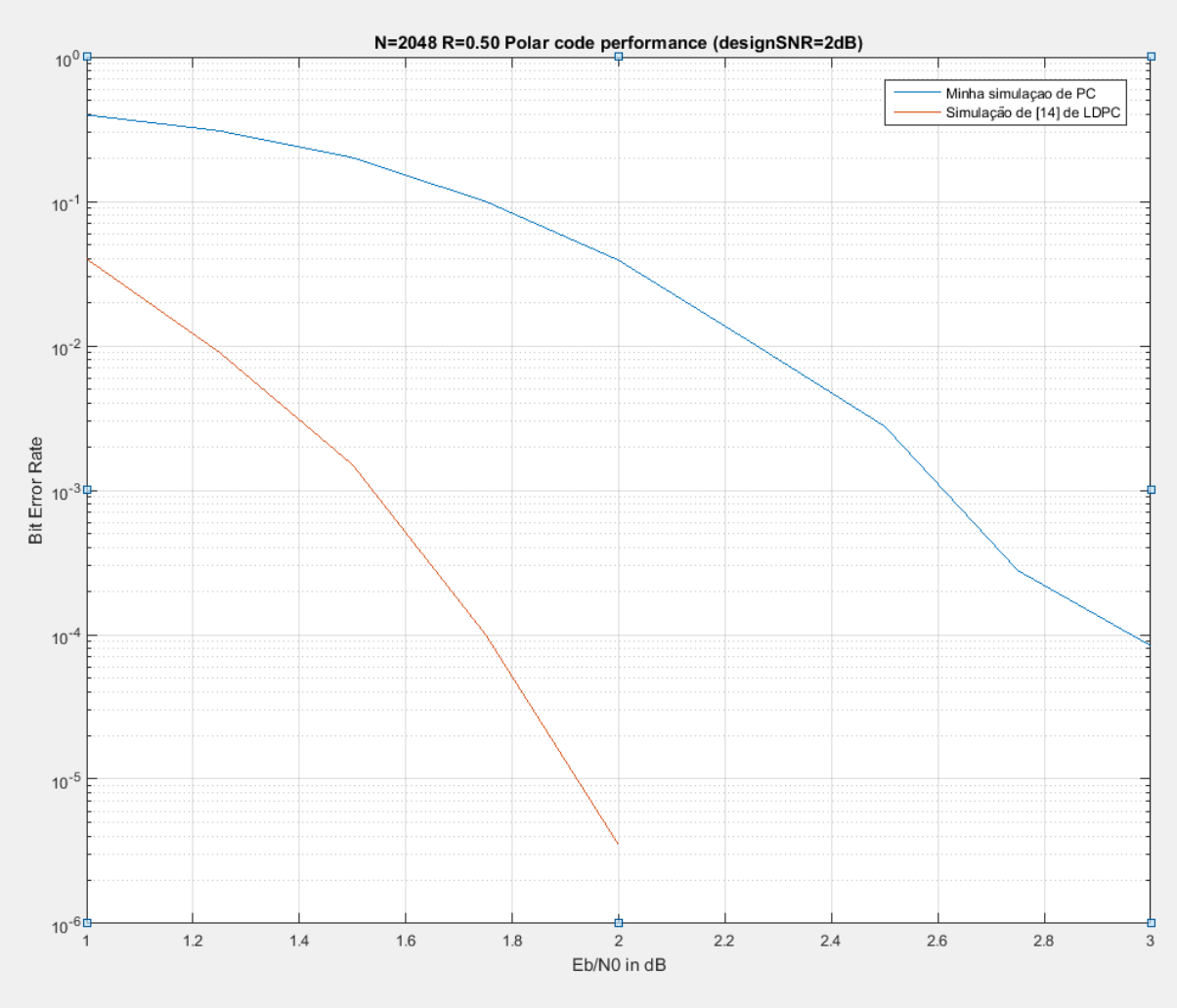


Figura 4.11: Comparação entre um código LDPC, padrão WiMAX, de comprimento N=2304 e taxa R=½, com os resultados de simulação retirados de [14], e um código polar de comprimento N=2048 e taxa R=½ e projetado para Eb/N0=2dB.

5 CONCLUSÃO

Neste trabalho, os conceitos primários de codificação polar foram revisados, incluindo polarização de canais, codificação e o algoritmo de cancelamento sucessivo (SC) para realizar a decodificação. Além disso, foi implementado em Matlab um código que reproduz os resultados de decodificação SC da literatura. Dessas simulações, é possível concluir que com o aumento do comprimento da palavra código um melhor desempenho é atingido e que a SNR projetada influencia diretamente nesses resultados.

Visto que o primeiro trabalho sobre códigos polares foi divulgado em 2009, por Arikan [1], diversas técnicas para para melhorar seu desempenho foram desenvolvidas, como decodificador de cancelamento sucessivo permutado, apresentado em [17], o decodificador de cancelamento sucessivo em lista (SCL) e o decodificador de cancelamento sucessivo em lista com verificação de redundância cíclica (CRC), apresentado em [14]. Com essas técnicas, o desempenho dos códigos polares se equiparou ao dos códigos LDPCs. Antes delas, ou seja, com a decodificação baseada em cancelamento sucessivo, os códigos polares possuiam desempenho inferior aos códigos LDPC, como apresentado na Figura 4.5.

Outro ponto interessante é que, por se tratar de uma nova técnica de codificação, ela foi pouco estudada no Brasil. Sendo assim, este trabalho de conclusão de curso pode servir de base para um primeiro contato com os códigos polares para pesquisadores, professores e estudantes brasileiros interessados no assunto.

Alguns possíveis trabalhos futuros podem ser um estudo aprofundado dos decodificadores em lista com verificação de redundância cíclica e da arquitetura do hardware de decodificadores de cancelamento sucessivo apresentado em [10].

REFERÊNCIAS

[1] Arikan, Erdal. "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels." *IEEE Transactions on Information Theory* 55.7 (2009): 3051-3073.

[2] Zehfuss, G. "Über eine gewisse Determinante." *Zeitschrift für Mathematik und Physik* 3.1858 (1858): 298-301.

[3] Elster, Anne Cathrine. "Fast bit-reversal algorithms." *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*. IEEE, 1989.

[4] Henry Pfister. A brief introduction to polar codes. http://pfister.ee.duke.edu/courses/ecen655/polar.pdf, 2014.

[5] Shannon, Claude E. "A mathematical theory of communication, Part I, Part II." *Bell Syst. Tech. J.* 27 (1948): 623-656.

[6] MacKay, David JC. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[7] Berrou, Claude, Alain Glavieux, and Punya Thitimajshima. "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1." *Communications, 1993. ICC'93 Geneva. Technical Program, Conference Record, IEEE International Conference on*. Vol. 2. IEEE, 1993.

[8] Gallager, Robert. "Low-density parity-check codes." *IRE Transactions on information theory* 8.1 (1962): 21-28.

[9] Richardson, Thomas J., Mohammad Amin Shokrollahi, and Rüdiger L. Urbanke. "Design of capacity-approaching irregular low-density parity-check codes." *IEEE transactions on information theory* 47.2 (2001): 619-637.

[10] Leroux, Camille, et al. "Hardware architectures for successive cancellation decoding of polar codes." *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011.

[11] Fossorier, Marc PC, Miodrag Mihaljevic, and Hideki Imai. "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation." *IEEE Transactions on communications* 47.5 (1999): 673-680.

[12] Dosio, Davide. *Polar codes for error correction: analysis and decoding algorithms*. Diss. 2016.

[13] Moon, Todd K. "Error correction coding." *Mathematical Methods and Algorithms. Jhon Wiley and Son* (2005).

[14] Tal, Ido, and Alexander Vardy. "List decoding of polar codes." *IEEE Transactions on Information Theory* 61.5 (2015): 2213-2226.

[15] Vangala, Harish, Emanuele Viterbo, and Yi Hong. "A comparative study of polar code constructions for the AWGN channel." *arXiv preprint arXiv:1501.02473* (2015).

[16] Vangala, Harish, Yi Hong, and Emanuele Viterbo. "Efficient algorithms for systematic polar encoding." *IEEE communications letters* 20.1 (2016): 17-20.

[17] Vangala, Harish, Emanuele Viterbo, and Yi Hong. "Permuted successive cancellation decoder for polar codes." *Information Theory and its Applications (ISITA), 2014 International Symposium on*. IEEE, 2014.